Lecture-2

# Data Link Layer (LLC and MAC)

**Dr. Md. Imdadul Islam**
Professor, Department of Computer Science and Engineering
Jahangirnagar University
*https://www.juniv.edu/teachers/imdad*

The function of logical link control layer (LLC) are: framing, flow control and error control.

# Framing

✓Framing refers to the process of partitioning bit stream into discrete units or blocks of data called frame. Framing enables sending and receiving machines to synchronize the transmission and reception of data because frames have detectable boundaries.

✓One common framing procedure involves inserting flag characters before and after the transmitting data message.


Fig.1 Frame format

- ✓ The header of a frame, normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits.

## Fixed-Size Framing

Frames can be of fixed or variable size. In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter (indication or symbol of boundary). An example of this type of framing is the ATM, which uses frames of fixed size called cells.

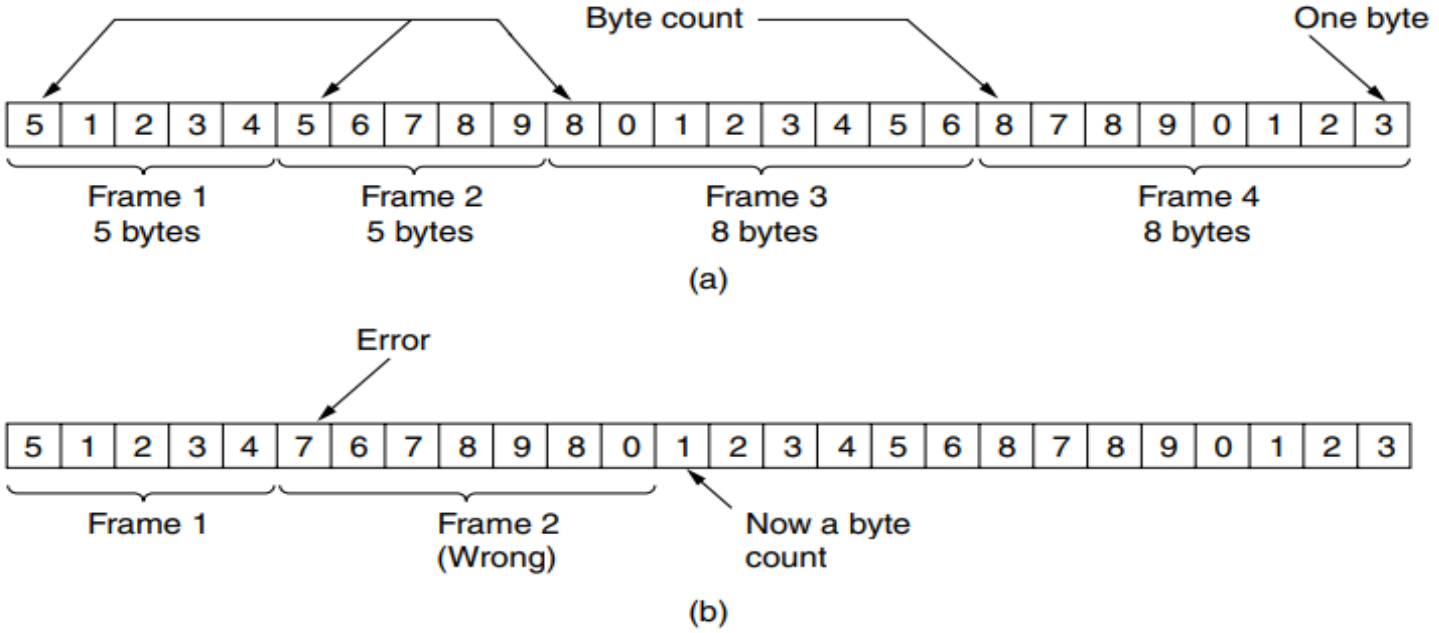
# Variable-Size Framing

Variable-size framing is prevalent in local area networks. In variable-size framing, we need a way to define the end of the frame and the beginning of the next. Three approaches were used for this purpose:

- **Byte count method**
- **Character-oriented approach (flag byte with byte stuffing)**
- **bit-oriented approach (Flag bits with bit stuffing)**
- **Physical layer coding violations**

# Byte count method

✓The first framing method (**Byte count method**) uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is. This technique is shown in Fig. a below for four frames of sizes 5, 5, 8, and 8 characters, respectively.



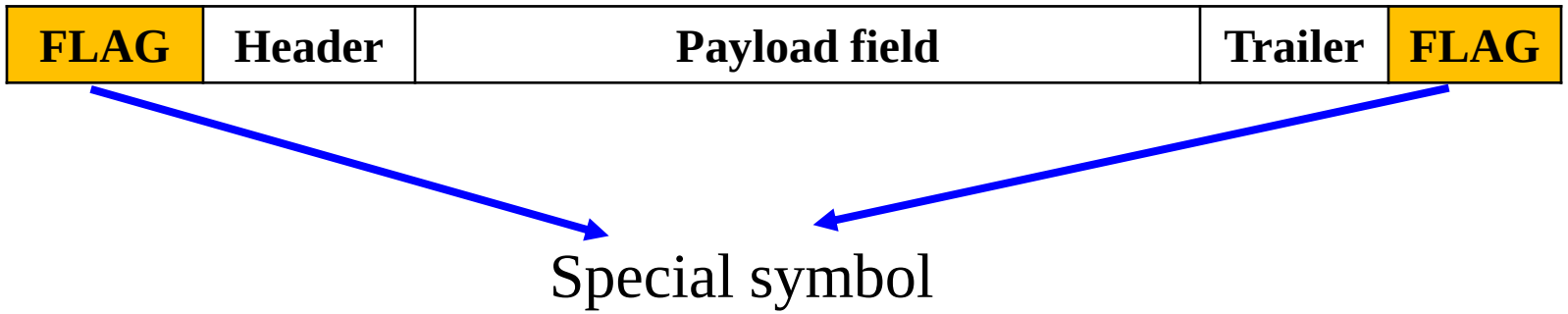

A character stream. (a) Without errors. (b) With one error.

✓The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the character count of 5 in the second frame of fig. b becomes a 7, the destination will get out of synchronization and will be unable to locate the start of the next frame

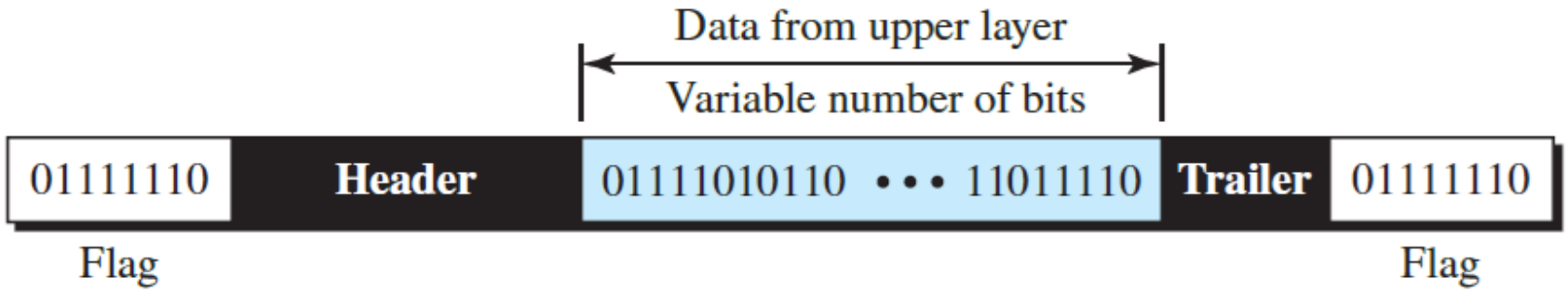
# Character-Oriented Protocols (flag byte with byte stuffing)

- ✓ In a character-oriented protocol (called **flag byte with byte stuffing)**, the component or element of header, data and the trailer composed of character or symbol (character of 8 bits or 1 byte) instead of bit.

- ✓ To separate one frame from the next, an 8-bit (1-byte) flag is added at the beginning and the end of a frame.

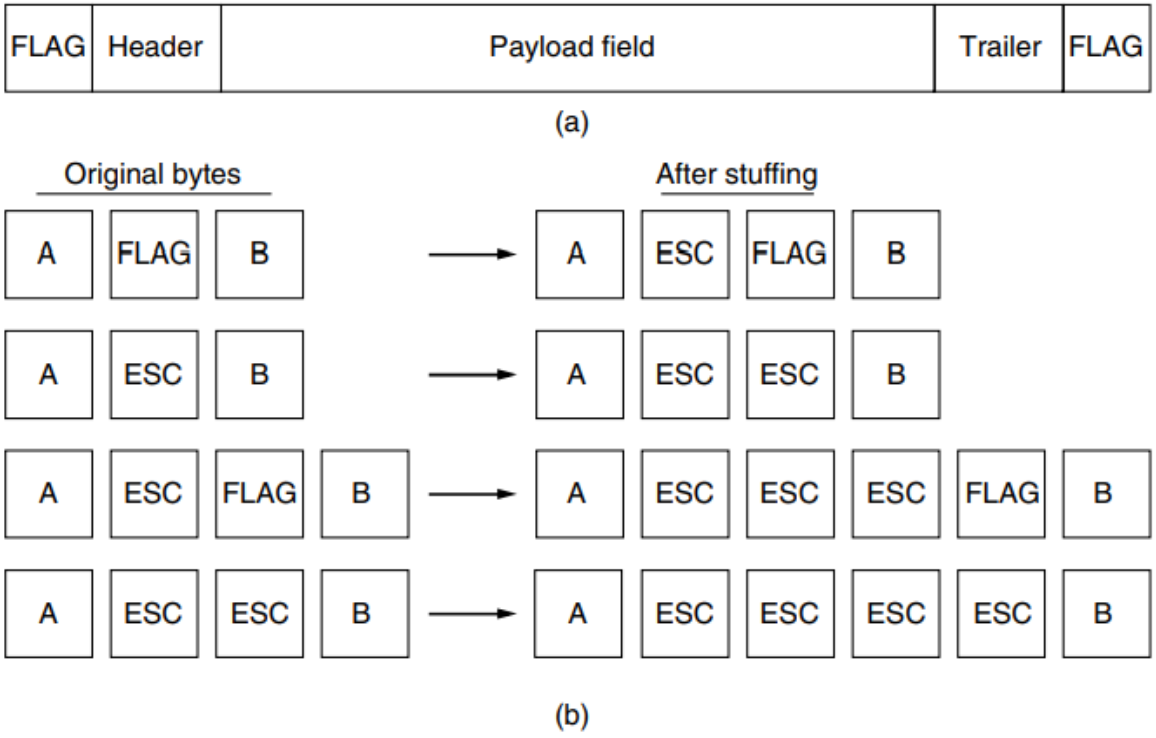


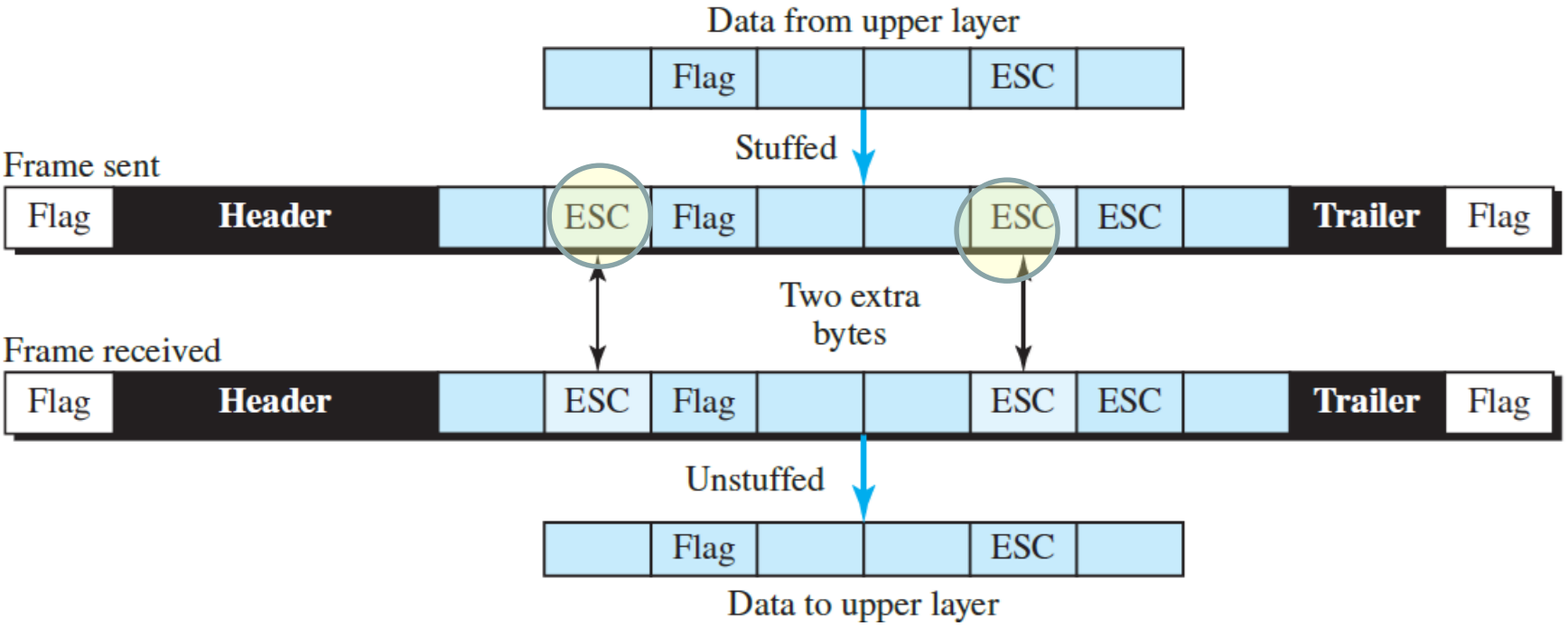
✓In this way, if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame. Two consecutive flag bytes indicate the end of one frame and start of the next one.



✓It may easily happen that the flag byte's bit pattern occurs in the data. This situation will usually interfere with the framing. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each 'accidental' flag byte in the data.

✓The data link layer on the receiving end removes the escape byte before the data are given to the network layer. This technique is called byte stuffing or character stuffing. Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it.






## Byte stuffing and unstuffing

The major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters or byte stuffing. Not all character codes use 8-bit characters for example UNICODE uses 16-bit or 32 bit characters, pixel of each colored image is 24 bits will conflict with 8-bit characters.

# Flag bits with bit stuffing

✓Therefore a third technique is developed to allow arbitrary sized characters called **bit-oriented protocol** (called **Flag bits with bit stuffing**). In this scheme, framing is done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size.

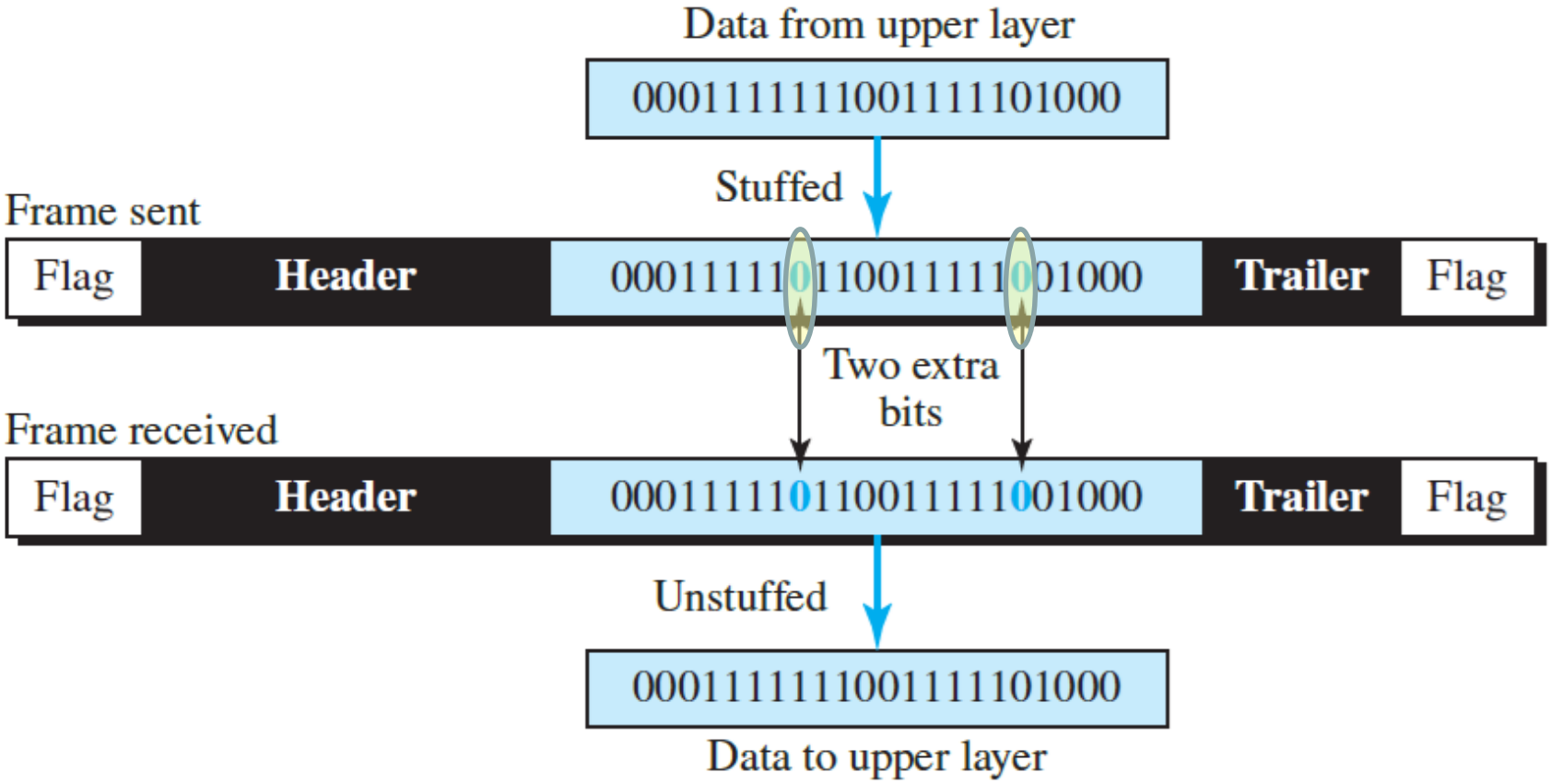
✓In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.

✓Each frame begins and ends with a special bit pattern, 01111110 (in fact, a flag byte). Whenever the sender's data link layer encounters five consecutive 1s in the data (payload), it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to byte stuffing.

✓When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically de-stuffs (i.e., deletes) the 0 bit.

✓For example, if the user data contain the flag pattern, **01111110**, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110.

Fig. below gives an example of bit stuffing.




Bit stuffing and unstuffing

- ✓ With both bit and byte stuffing, a side effect is that the length of a frame now depends on the contents of the data it carries.
- ✓ For instance, if there are no flag bytes in the data, 100 bytes might be carried in a frame of roughly 100 bytes (best case).
- ✓ If, however, the data consists solely of flag bytes, each flag byte will be escaped and the frame will become roughly 200 bytes long (worst case). (byte stuffing case)
- ✓ With bit stuffing, the increase would be roughly 12.5% as 1 bit is added to every byte. (bit stuffing case)

## Physical layer coding violations

The fourth method of framing is to use a shortcut from the physical layer, where pulse sequence of special line code (shape of pulse) which is not used inside data pulse of a frame can be used as the head and tail of the frame.

# Error Control

The term error control refers to the process of guaranteeing reliable data delivery. Two basic strategies exist for dealing with error.

- The first method, *error correction through retransmission*, involves providing enough information in the data stream so the receiving node can detect an error. Once an error is detected the receiving node can then request the sender to retransmit that unit data.

- The second method is *autonomous error correction*, involves proving redundant information in the data stream so the destination node can both detect and correct any error autonomously.

❑Error control using retransmission involves the use of acknowledgements the receiving node provides the sending node with feedback about the frames it has received. A positive acknowledgement means a frame was received correctly; a negative acknowledgement (or no acknowledgement) implies a frame was not received correctly. Negative acknowledgements imply that the sending node needs to retransmit the frame.

❑To guard against the possibility of lost or destroyed (no error) frames, the data link layer also supports timers. If a frame or acknowledgement is lost, the sending node's time limit eventually expires, altering it to retransmit the frame.

❑To guard a destination node accepting duplicate frames, outgoing frames are assigned sequence number, which enable a destination node to distinguish between retransmitted and the original one.

❑The concept of error detection, acknowledgement, and retransmission are collectively referred to as *automatic repeat request* (ARQ). Again when the receiver corrects the frame called *forward error correction* (FEC).

# Error Detection Coding

## The Cyclic Redundancy Check (CRC)

- ✓ The major goal in designing error detection algorithms is to maximize the probability of detecting errors (protect the undetected error) using only a small number of redundant bits. Cyclic redundancy checks use some mathematics to achieve this goal.

- ✓ For CRC coding we need a basic mathematics of 'bit sequence to polynomial conversion'. For example, an 8-bit message consisting of the bits, $M = 10011010$ corresponds to the polynomial,

$$M(x) = 1\times x^7 + 0\times x^6 + 0\times x^5 + 1\times x^4 + 1\times x^3 + 0\times x^2 + 1\times x^1 + 0\times x^0$$
$$= x^7 + x^4 + x^3 + x$$

Example-1

$M = 1\ 0\ 0\ 1\ 0\ 0\ 1 \leftrightarrow M(x) = x^6 + x^3 + 1$

$$S(x) = a_0 + a_1 x + a_2 x^2 + \ldots\ \ldots\ \ldots + a_n x^n$$

For the purposes of calculating a CRC, a sender and receiver have to agree on a divisor or generator polynomial, $G(x)$. $G(x)$ is a polynomial of degree $k$. For example, $G(x) = x^3 + x^2 + 1$. In this case, $k = 3$.

**Common CRC polynomials**

| CRC | $G(x)$ |
|---|---|
| CRC-8 | $x^8 + x^2 + x + 1$ |
| CRC-10 | $x^{10} + x^9 + x^5 + x^4 + x + 1$ |
| CRC-12 | $x^{12} + x^{11} + x^3 + x^2 + 1$ |
| CRC-16 | $x^{16} + x^{15} + x^2 + 1$ |
| CRC-CCITT | $x^{16} + x^{12} + x^5 + 1$ |
| CRC-32 | $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ |

# Steps of determining transmitted polynomial

We wanted to create a polynomial $T(x)$ for transmission that is derived from the original message $M(x)$. Transmitted bit string $T$ is $r$ bits longer than $M$, and $T(x)$ is exactly divisible by $G(x)$, where $r$ is the degree of $G(x)$.

We can derive $T(x)$ from $M(x)$ in the following way:

1. Multiply the message polynomial $M(x)$ by $x^r$, which is an equivalent polynomial of a sequence derived by adding $r$ zeros at the end of the message. We call this zero-extended message $x^rM(x)$.

2. Divide $x^rM(x)$ by $G(x)$ and find the remainder $R(x)$.

3. Subtract the remainder $R(x)$ from the dividend $x^rM(x)$ to get $T(x)$.

It is obvious that $T(x)$ is exactly divisible by $G(x)$.

**Example-1:**

Given, Message bit string $M$ = 1 0 1 0 0 0 1 1 0 1 (10 bits)

Generator bit string $G$ = 1 1 0 1 0 1 (6 bits)

Determine remainder polynomial $R(x)$ and transmitted polynomial $T(x)$.

Ans.

Given, Message bit string, $M$ = 1 0 1 0 0 0 1 1 0 1 (10 bits)

Generator bit string, $G$ = 1 1 0 1 0 1 (6 bits)

Determine $R(x)$ and $T(x)$.

$$\therefore M(x) = x^9 + x^7 + x^3 + x^2 + x^0 \qquad G(x) = x^5 + x^4 + x^2 + x^0$$

Degree of $G(x)$, $r$ = 5.

Message after appending 5 zeros becomes,

1 0 1 0 0 0 1 1 0 1 0 0 0 0 0

The corresponding polynomial, $x^5 M(x) = x^{14} + x^{12} + x^8 + x^7 + x^5$

Dividend

Divisor $x^5 + x^4 + x^2 + 1$

$x^{14} + x^{12} + x^8 + x^7 + x^5$

$x^{14} + x^{13} + x^{11} + x^9$

Quotient $x^9 + x^8 + x^6 + x^4 + x^2 + x$

$x^{13} + x^{12} + x^{11} + x^9 + x^8 + x^7 + x^5$

$x^{13} + x^{12} + x^{10} + x^8$

$x^{11} + x^{10} + x^9 + x^7 + x^5$

$x^{11} + x^{10} + x^8 + x^6$

$x^9 + x^8 + x^7 + x^6 + x^5$

$x^9 + x^8 + x^6 + x^4$

$x^7 + x^5 + x^4$

$x^7 + x^6 + x^4 + x^2$

$x^6 + x^5 + x^2$

$x^6 + x^5 + x^3 + x$

$x^3 + x^2 + x$ Remainder

$G(x)$ | $x^5 M(x)$ | $Q(x)$

$R(x)$

12 | 79 / 72 | 6

7

Modulo 2 operation

$\therefore R(x) = x^3 + x^2 + x \leftrightarrow 0\ 1\ 1\ 1\ 0$

$\therefore T(x) = x^5 M(x) - R(x)$

$= x^{14} + x^{12} + x^8 + x^7 + x^5 + x^3 + x^2 + x$

Message string after appending 5 zeros, **1 0 1 0 0 0 1 1 0 1 0 0 0 0 0**

```
1 1 0 1 0 1 ) 1 0 1 0 0 0 1 1 0 1 0 0 0 0 0 ( 1 1 0 1 0 1 0 1 1 0
              1 1 0 1 0 1
              ___________
                1 1 1 0 1 1
                1 1 0 1 0 1
                ___________
                  1 1 1 0 1 0
                  1 1 0 1 0 1
                  ___________
                    1 1 1 1 1 0
                    1 1 0 1 0 1
                    ___________
                      1 0 1 1 0 0
                      1 1 0 1 0 1
                      ___________
                        1 1 0 0 1 0
                        1 1 0 1 0 1
                        ___________
                          0 1 1 1 0 (R)
```

```
1 0 1 0 0 0 1 1 0 1 0 0 0 0 0
            +         0 1 1 1 0
_____________________________________
T = 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0
```

# Transmitted bit sequences and polynomial

$$T(x) = x^{14} + x^{12} + x^8 + x^7 + x^5 + x^3 + x^2 + x$$

$$T = \mathbf{1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0}$$

$$R(x) = x^3 + x^2 + x \leftrightarrow 0\ \ 1\ \ 1\ \ 1\ \ 0 = R$$

$$G(x) = x^5 + x^4 + x^2 + x^0$$

$$\frac{T(x)}{G(x)} = Q(x) = x^9 + x^8 + x^6 + x^4 + x^2 + x$$

The received polynomial

$$R_e(x) = T(x) + E(x)$$

$M = 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1$

# Error Polynomial

Case-I

$$T = \mathbf{1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0}$$

$$T(x) = x^{14} + x^{12} + x^{8} + x^{7} + x^{5} + x^{3} + x^{2} + x$$

If the received sequence is:

$$Re = \mathbf{1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0}$$

$$\mathrm{Re}(x) = x^{14} + x^{12} + x^{10} + x^{8} + x^{7} + x^{5} + x^{3} + x^{2} + x$$

$$= T(x) + E(x)$$

Where error polynomial, $E(x) = x^{10}$

# Error Polynomial

Case-II

$$T = \mathbf{1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0}$$

$$T(x) = x^{14} + x^{12} + x^{8} + x^{7} + x^{5} + x^{3} + x^{2} + x$$

If the received sequence is:

$$Re = \mathbf{1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0}$$

$$\mathrm{Re}(x) = x^{14} + x^{12} + x^{7} + x^{5} + x^{3} + x^{2} + x$$

$$= T(x) + E(x)$$

Where error polynomial, $E(x) = x^8$

# Error Polynomial

Case-III

$$T = \mathbf{1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0}$$

$$T(x) = x^{14} + x^{12} + x^8 + x^7 + x^5 + x^3 + x^2 + x$$

If the received sequence is:

$$Re = \mathbf{1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0}$$

$$\mathrm{Re}(x) = x^{14} + x^{10} + x^8 + x^7 + x^5 + x^2 + x$$

$$= T(x) + E(x)$$

Where error polynomial, $E(x) = x^{12} + x^{10} + x^3$

If there is some error at detecting end then polynomial of the received string will be, $T(x) + E(x)$ where $E(x)$ is arises from error.

Now

$$\frac{R_e(x)}{G(x)} = \frac{T(x) + E(x)}{G(x)} = \frac{T(x)}{G(x)} + \frac{E(x)}{G(x)}$$

$$= Q(x) + \frac{E(x)}{G(x)}$$

## Choice of $G(x)$:

Here if $E(x)$ is divisible by $G(x)$ then error can't be detected. To combat the situation, choosing of $G(x)$ has some criteria like below.

1. One common type of error is a single-bit error, which can be expressed as $E(x) = x^i$ when it affects bit position $i$. If we select $G(x)$ such that the first and the last term are nonzero, then we already have a two-term polynomial that cannot divide evenly into the one term $E(x)$.

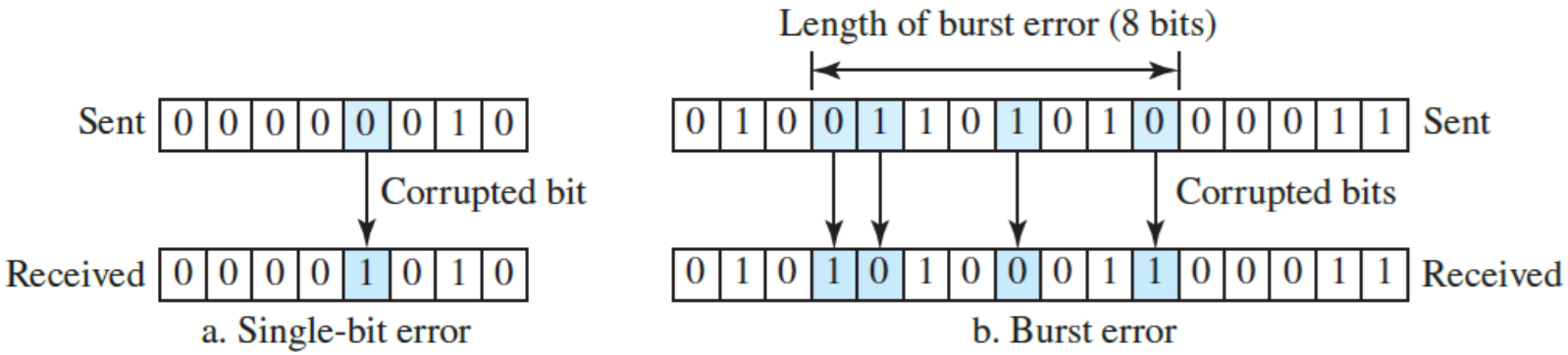
2. For two bit error, $E(x) = x^m + x^r;\ m > r$

$= x^r(x^{m-r} + 1) = x^r(x^k + 1)$

In this case $G(x)$ has to be chosen such that it does not divide $x^k + 1$. For example if $G(x) = x^{15} + x^{14} + 1$ then $G(x)$ is unable to divide $x^k + 1$ for $k \leq 32{,}768$.

3. For odd number of bits in error then $E(x)$ will contains odd number of terms. For 3 bit error, $E(x) = x^5 + x^2 + 1$. If any polynomial has odd number of terms then it does not have any factor like $(x+1)$. If $G(x)$ has a factor of $(x+1)$ it would be unable to divide $E(x)$.

The term **burst error** means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.




- ✓ A burst error is more likely to occur than a single-bit error because the duration of the noise signal is normally longer than the duration of one bit, which means that when noise affects data, it affects a set of bits.

- ✓ The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 kbps, a noise of 1/100 s can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

$$G(x) = x^5 + x^4 + x^2 + x^0, r = 5$$

4. For burst error of length $k$ then

$$E(x) = x^{k+i-1} + x^{k+i-2} + \ldots + x^{i+1} + x^i$$
$$= x^i (x^{k-1} + x^{k-2} + \ldots + 1)$$

If the degree of $G(x)$ is $r$ and $r \geq k\text{-}1$ then $G(x)$ is simply unable to divide $E(x)$ i.e. error is detected.

- ✓ For burst error of the size $k = r+1$ is detected with probability, $P = 1 - (1/2)^{r-1}$
- ✓ For burst error of the size $k > r+1$ is detected with probability, $P = 1 - (1/2)^r$

5. If the error sequence and generator are identical then the CRC code will failed to detect error. The probability of such case,

$$P = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \ldots (r+1) \text{ th term}$$

$= (1/2)^{r+1,}$ which is very small

$E(x) = G(x)$

$G = 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1$
$E = 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1$

# Dividing Circuit Design

Division operation is done by a sequential digital circuit, consists of X-OR gates and a shift resister. Design procedure includes three steps.

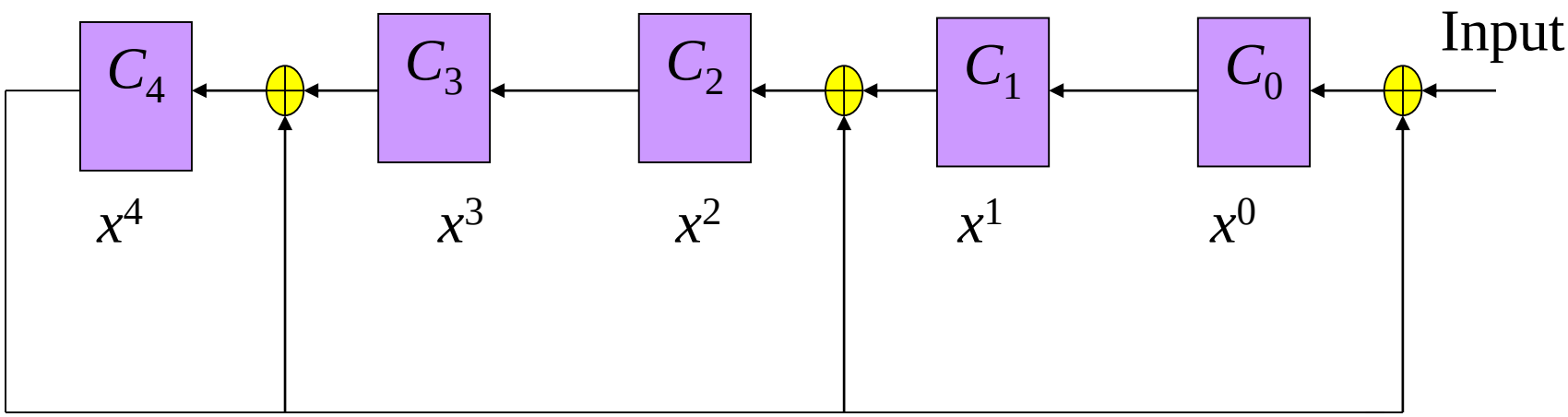
❖Number flip-flop of shift resister is equal to highest order of $G(x)$ i.e. $r$.

❖Maximum number X-OR gates could be $r$.

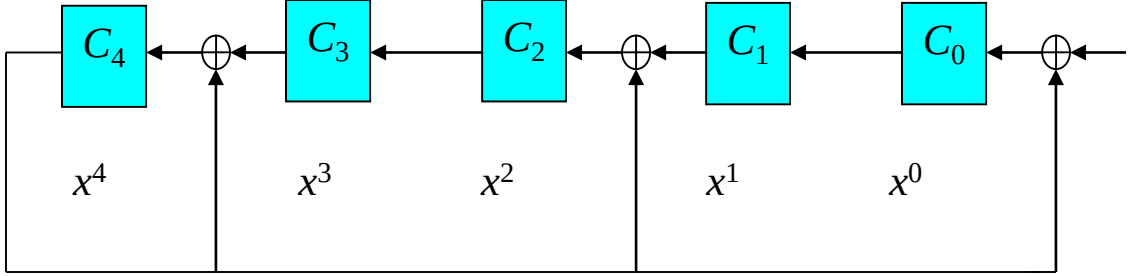
❖Presence or absence of X-OR gate in the circuit after each flip-flop depends upon presence / absence of terms of $G(x)$ excluding $x^r$.

For example, $G(x) = x^5 + x^4 + x^2 + 1$

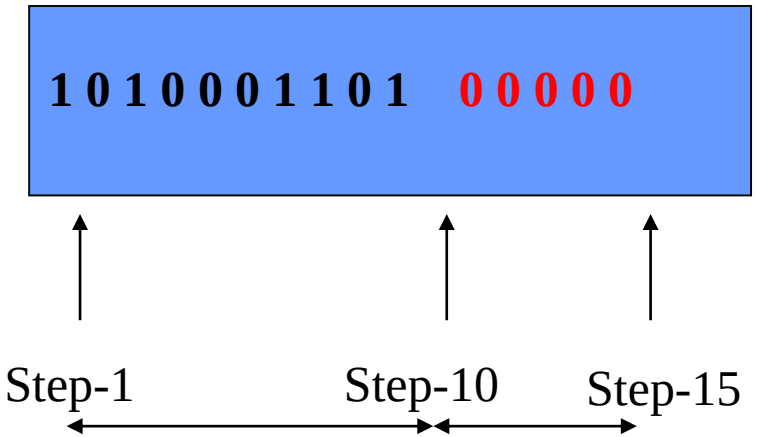
Number of FF is 5 and X-OR gate is 3 (exclusion $x^5$ of from $G(x)$ remains three terms). The complete circuit is drawn in fig. below.




Dividing circuit of $G(x) = x^5+x^4+x^2+1$




| Steps | $C_4$ | $C_3$ | $C_2$ | $C_1$ | $C_0$ |
|---|---|---|---|---|---|
| Initial | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 1 | 0 |
| 5 | 1 | 0 | 1 | 0 | 0 |
| 6 | 1 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 | 0 | 1 |
| 9 | 0 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 | 1 |
| 12 | 1 | 0 | 1 | 1 | 0 |
| 13 | 1 | 1 | 0 | 0 | 1 |
| 14 | 0 | 0 | 1 | 1 | 1 |
| 15 | **0** | **1** | **1** | **1** | **0** |




The content of the shift resister at 15th step gives the remainder of the division.

**Remainder**

```
%Matlab code
msg=[1 0 1 0 0 0 1 1 0 1]';
gen = crc.generator('Polynomial' ,[1 1 0 1 0 1]);
%here generator  = 1 1 0 1 0 1
encoded = generate(gen, msg);
det = crc.detector('Polynomial',[1 1 0 1 0 1]);
%here generator  = 1 1 0 1 0 1 is put gain
[outdata error] = detect(det, encoded);
noErrors = isequal(msg, outdata)
%Result will be logic 1 if msg and outdata are equal
```

**Msg' =** 1 0 1 0 0 0 1 1 0 1

**outdata' =** 1 0 1 0 0 0 1 1 0 1

**encoded' =** 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0

Q. An information source generates message bit string, $M$ = 1 0 0 0 1 0 1 1 0 1 0 1 (12 bits)
Generator bit string, $G$ = 1 0 0 1 0 1 1 (7 bits)
i) Determine the polynomials: $R(x)$ and $T(x)$.
ii) Design the divisor circuit.
iii) How to avoid ambiguity of single bit, 2 bits and odd number of bit error, burst error and the problem of E(x)= G(x)?

# Flow Control

- ✓ The various stations in a network may operate at **different speeds**. One of the tasks of the data link layer is to ensure that slow devices are not swamped with data from fast devices.

- ✓ The sending station must not send frames at a rate faster than the receiving station can absorb them.

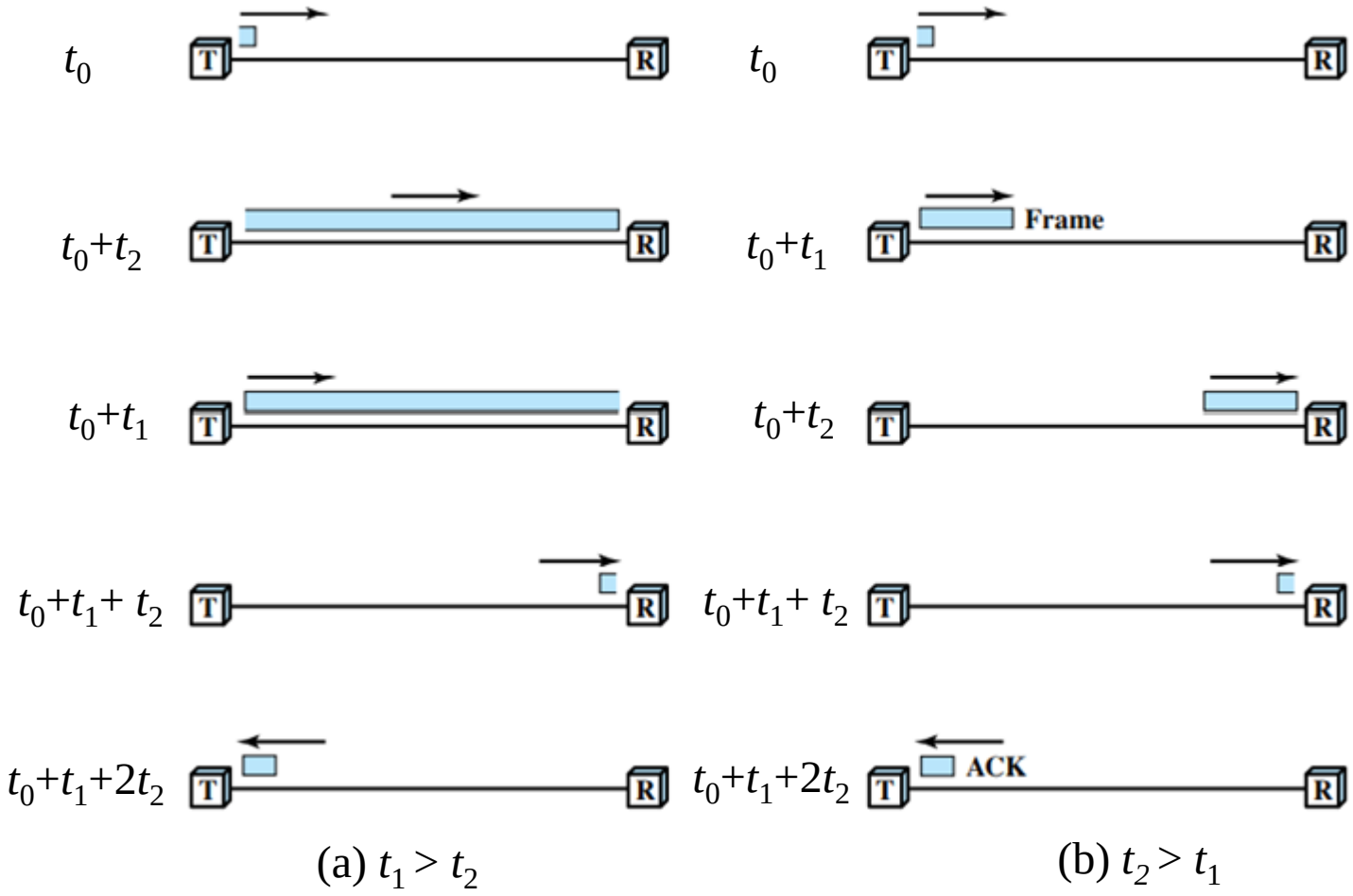
- ✓ Flow control refers to the regulating of the rate of data flow from one device to another so that the receiver has enough time to consume the data in its receive buffer, before it overflows.

# Stop-and-Wait Flow Control

- ✓ The simplest form of flow control, known as stop-and-wait flow control.
- ✓ A source entity transmits a frame. After the destination entity receives the frame, it indicates its willingness to accept another frame by sending back an **acknowledgment** to the frame just received.
- ✓ The source must wait until it receives the **acknowledgment** before sending the next frame.
- ✓ The destination can thus stop the flow of data simply by withholding acknowledgment.

- ✓ This protocol works fine when a message is sent in a few large frames but its performance is very poor for small frames.
- ✓ However, it is often needed by a source to break up a large block of data into smaller blocks to accommodate the receiver's limited buffer.
- ✓ Small frame sizes also facilitate faster error detection/correction and reduce the amount of data requires retransmission in the event of error detection.
- ✓ Small frame is also necessary to reduce network congestion.
- ✓ For this case of small frame the line is always **underutilized**.

Let, the transmission time (the time it takes for a node to transmit a frame) is $t_1$, and the propagation delay (the time it takes for a bit to travel from sender to receiver) is expressed as the variable $t_2$.



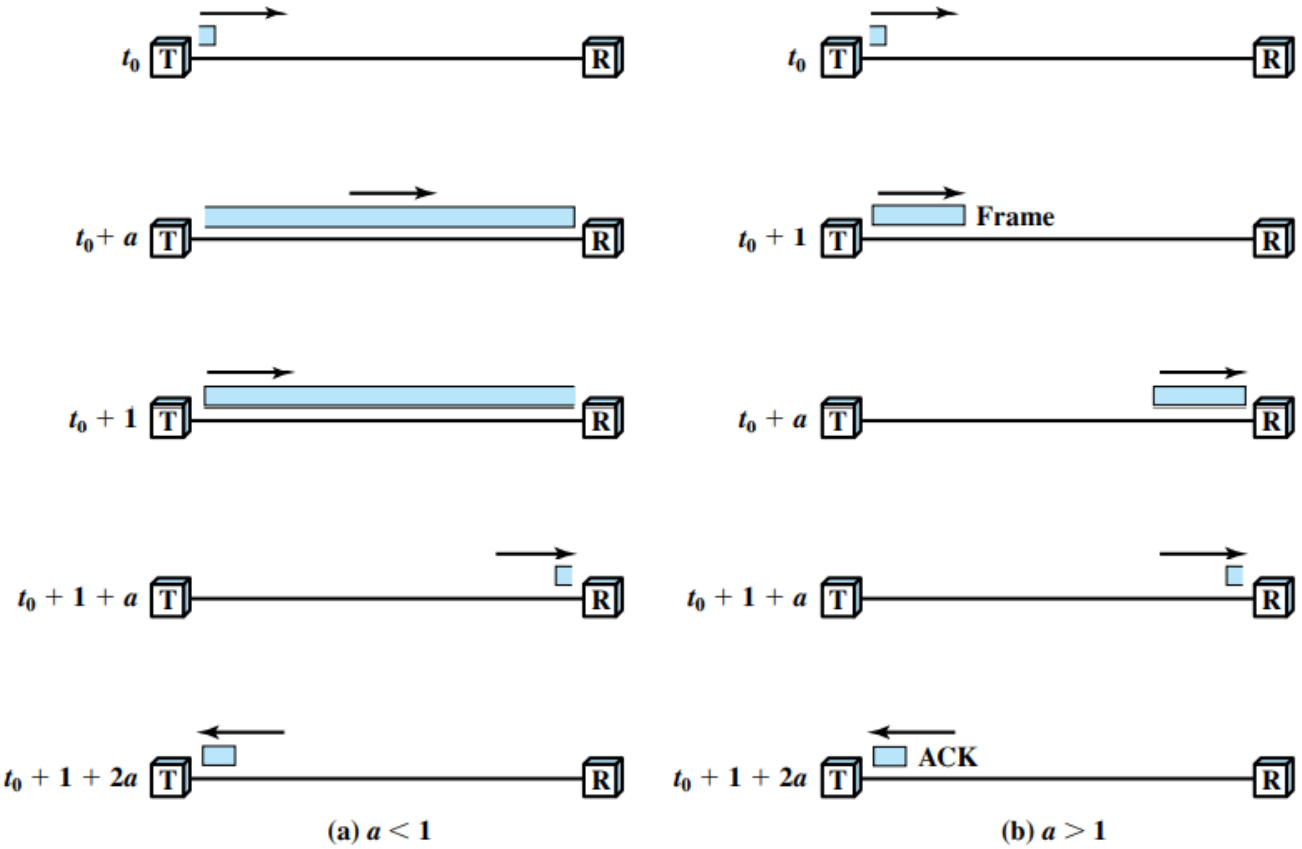

(a) $t_1 > t_2$

(b) $t_2 > t_1$

Transmission time is larger than propagation delay

Transmission time is smaller than propagation delay

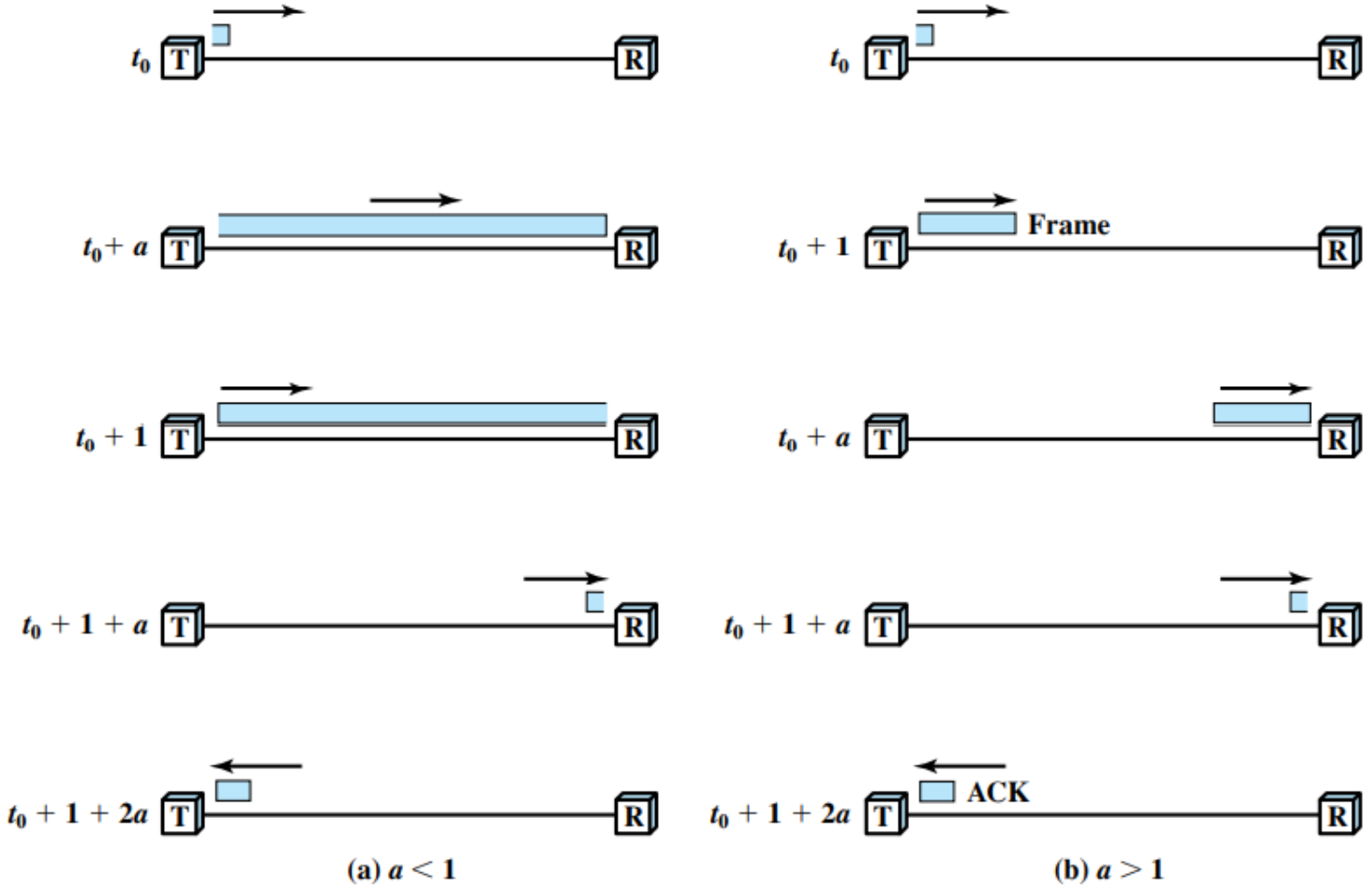
Let, the transmission time (the time it takes for a node to transmit a frame) is normalized to 1, and the propagation delay (the time it takes for a bit to travel from sender to receiver) is expressed as the variable $a$. (originally transmission time is $t_1$ and propagation delay is $t_2$, therefore their normalize values will be, $1 = t_1/t_1$, $a = t_2/t_1$.




Transmission time is larger than propagation delay

Transmission time is smaller than propagation delay

In both cases, the first four snapshots show the process of transmitting a frame containing data, and the last snapshot shows the return of a small acknowledgment frame, where transmission time = 1 and propagation time = *a*.




For $a > 1$ the line is always underutilized

# Sliding Window Flow Control

✓An enhancement to the stop-wait protocol is the sliding window concept, which improves data flow by having the receiver inform the sender of its available buffer space.

✓Doing so it enables the sender to transmit frames continuously without having to wait for acknowledgements to these frames as long as the number of frames sent does not overflow the receiver's buffers.

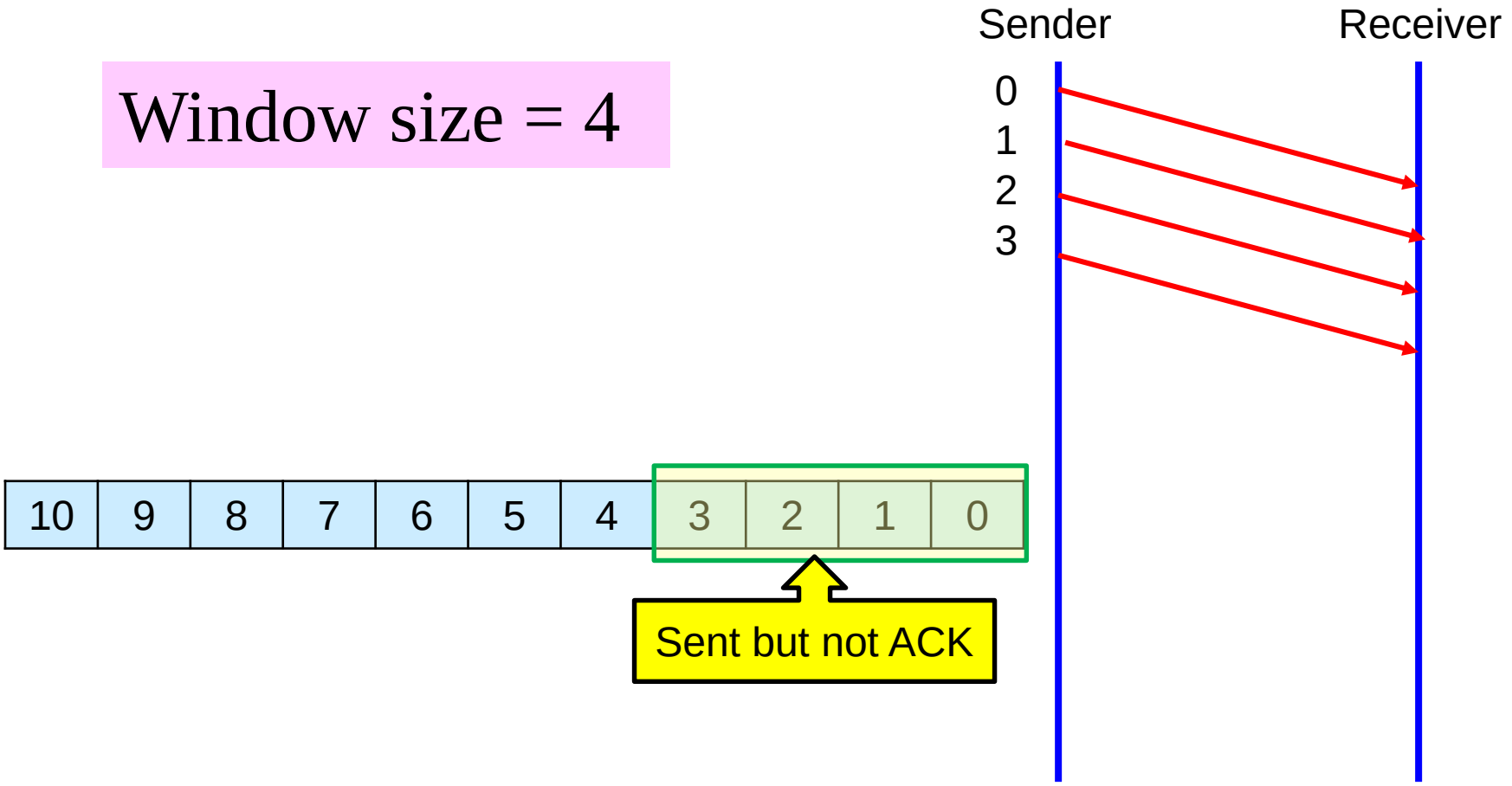
✓The sliding window concept is implemented by requiring the sender to sequentially number each data frame it sends.

✓Using this number the sender and receiver now maintain information about the number of frames they can respectively send or receive.

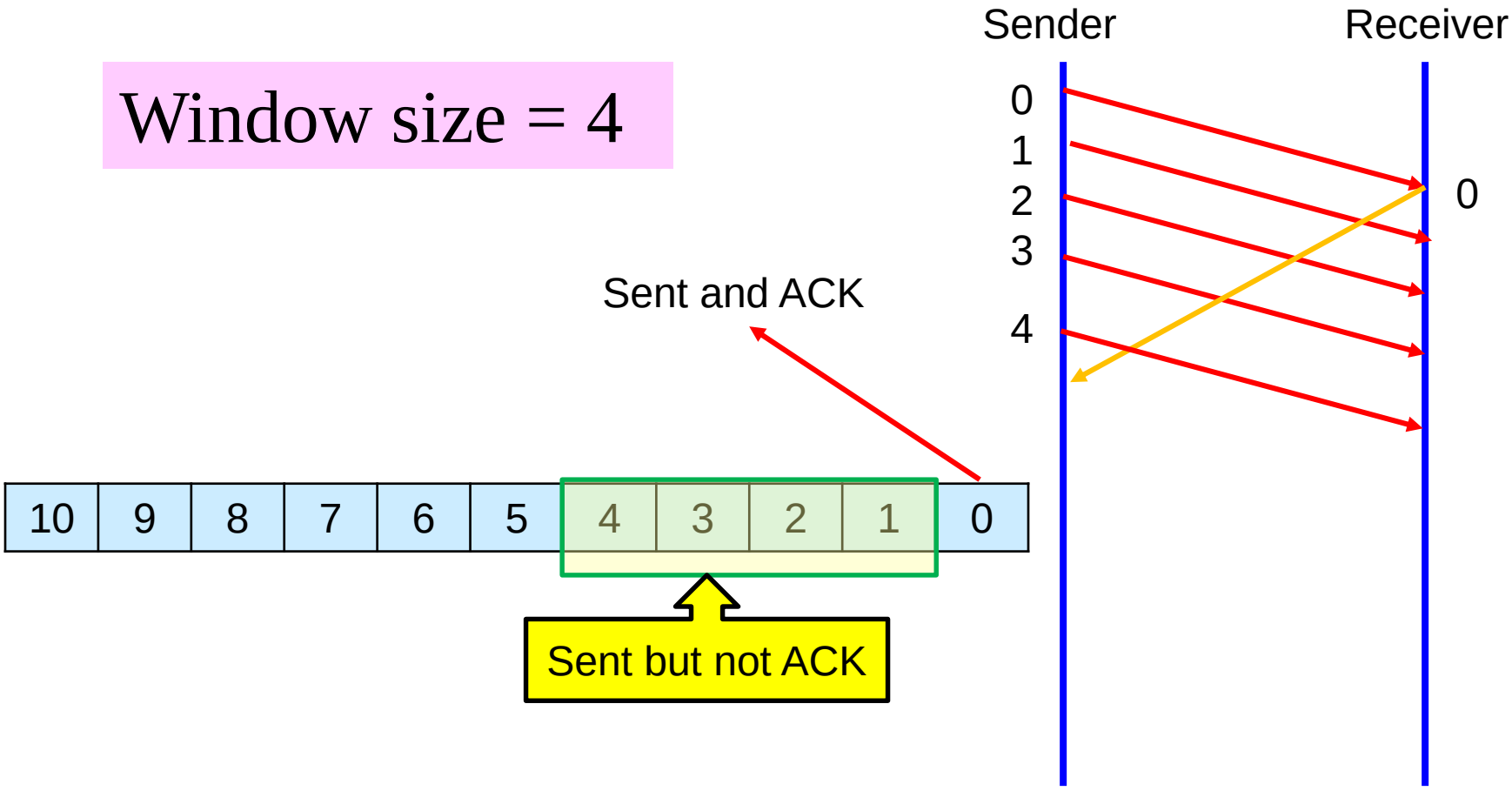
✓Flow-control protocols based on this concept are called **sliding window** protocol.

Consider a situation of fixed window size of 4. Every frame is numbered like 0, 1, 2, …10.

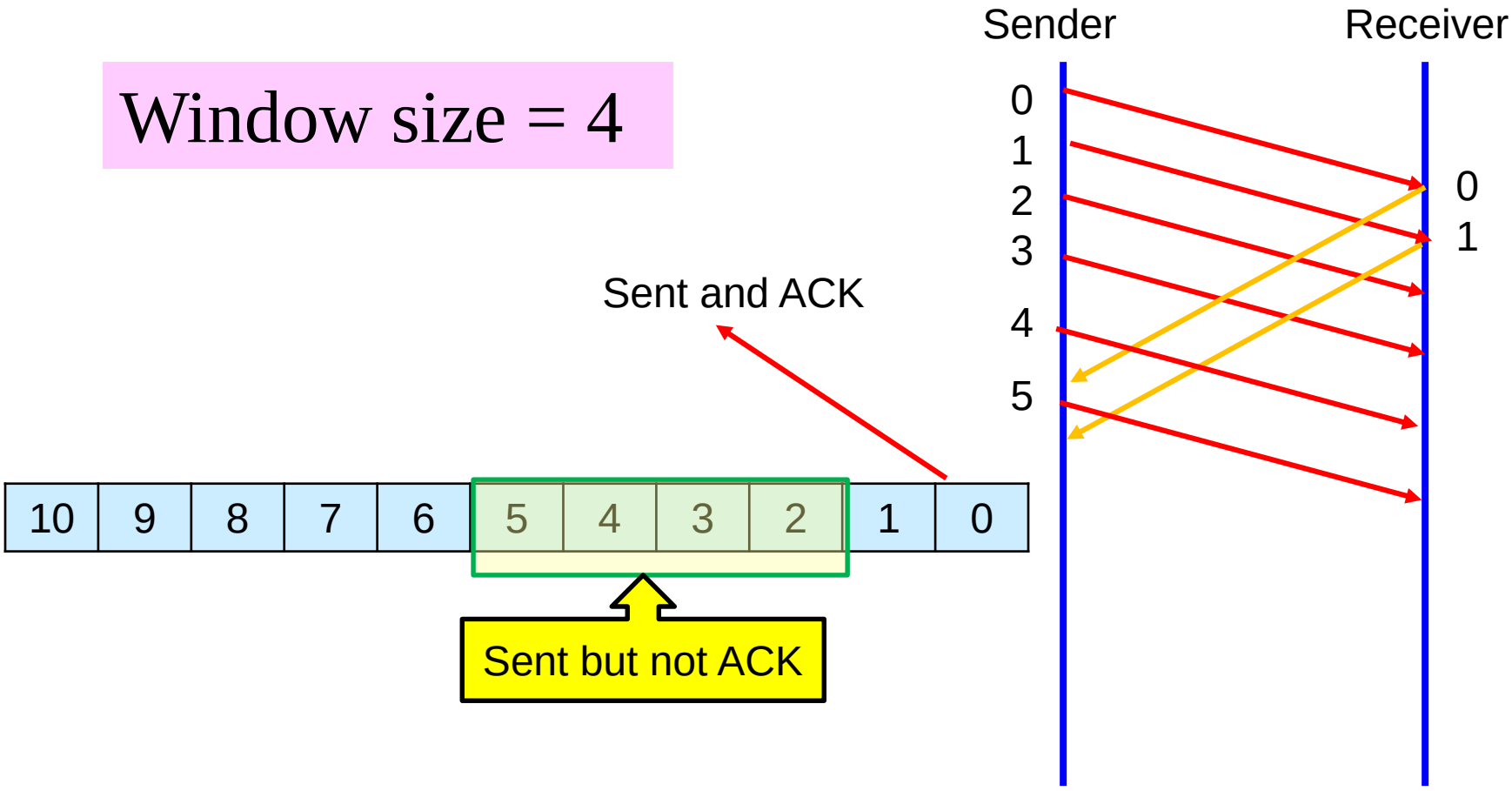
Window size = 4



# Window size = 4



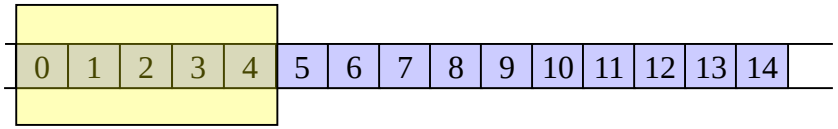
# Window size = 4



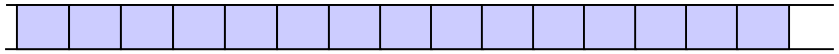
# Situation of Variable Window

## Step-1

**Sender (Host – A)**



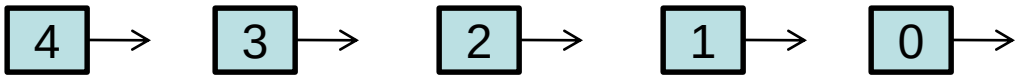

**Receiver (Host – B)**

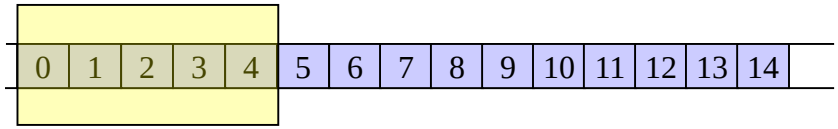


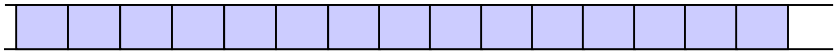
Initialization: A's send window is five

Initialization: B's receive window is five

## Step-2

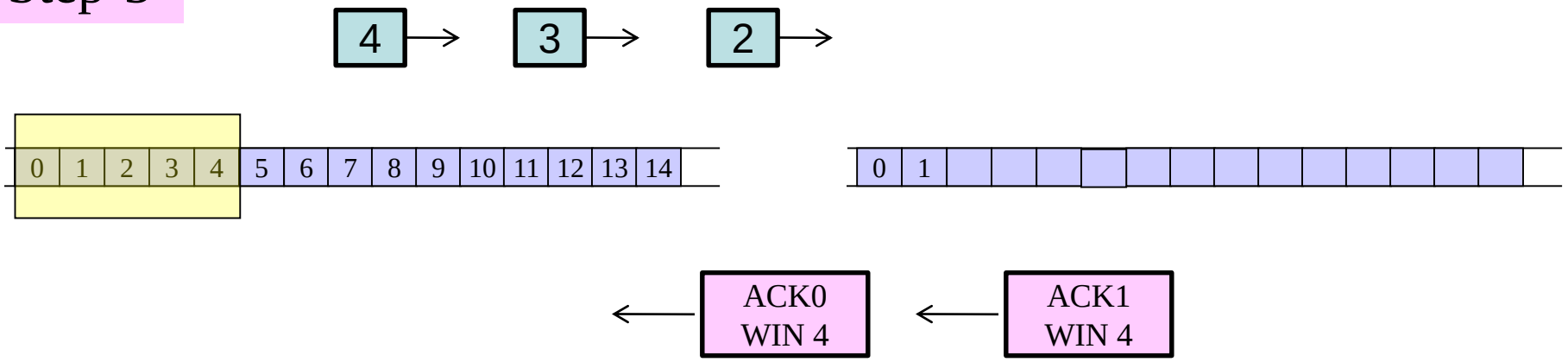


A transmits the first 5 frames

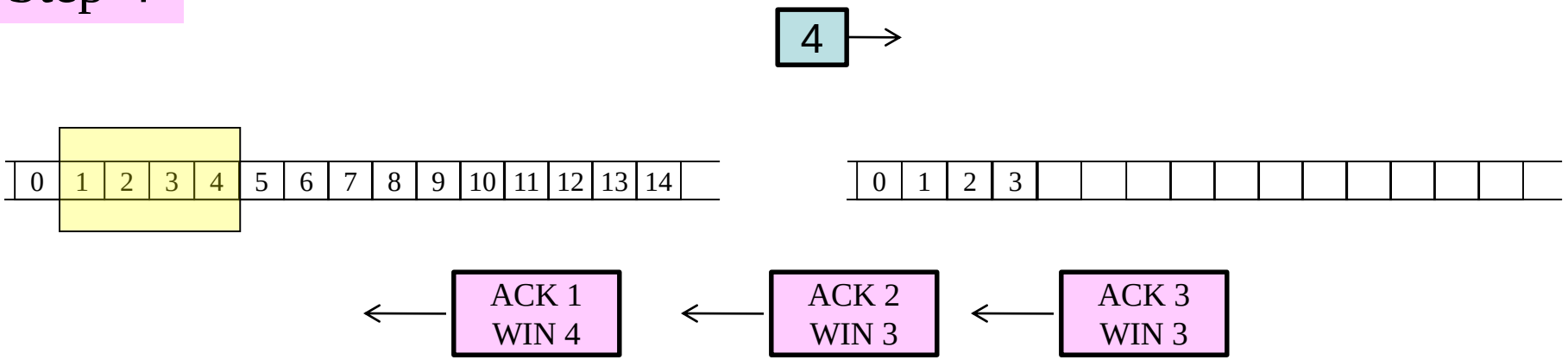
## Step-3




A has not received any ACK so it keeps the window size 5 like before

B Receives frame 0 and 1 and sends the corresponding ACK. Each ACK carries a window size of 4. Thus once A receives the ACK for frame 0, it must change its widow size to 4.

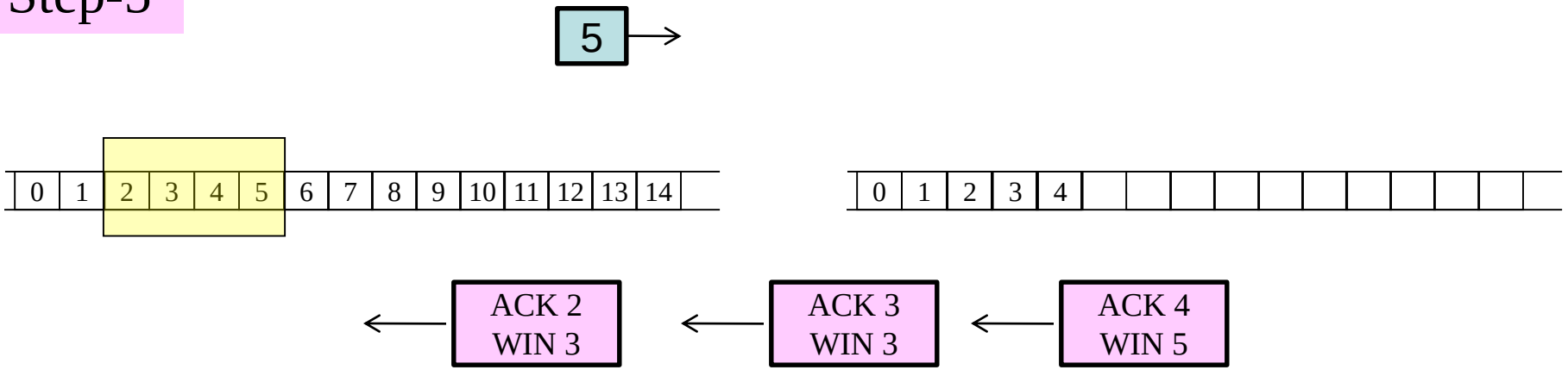
# Step-4




A has ACK of frame 0 and advances the left side of its window by one. However, frame 0's ACK had the window size 4 of 4 therefore A cannot advance the right side of its window.

B Receives frame 2 and 3 and sends the corresponding ACK. Each ACK carries a window size of 3. Thus once A receives the ACK, it must change its widow size to 3.

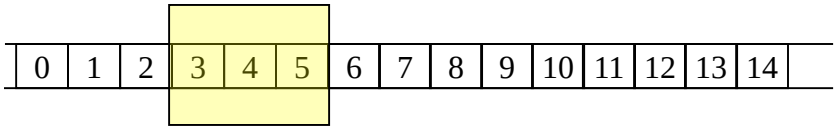
# Step-5

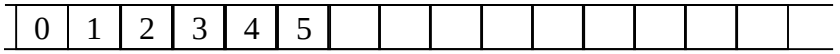


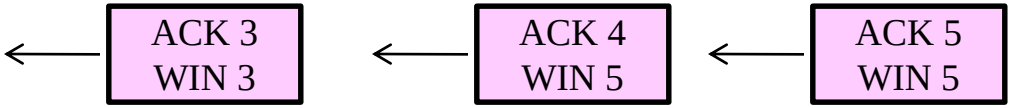

A has ACK of frame 1 and advances the left side of its window by one. However, frame 1's ACK had the window size of 4 therefore A can now advance the right side of its window by one and send frame 5

B Receives frame 4 and sends the corresponding ACK with window size of 5. Thus once A receives the ACK, it must change its widow size to 5.

# Step-6



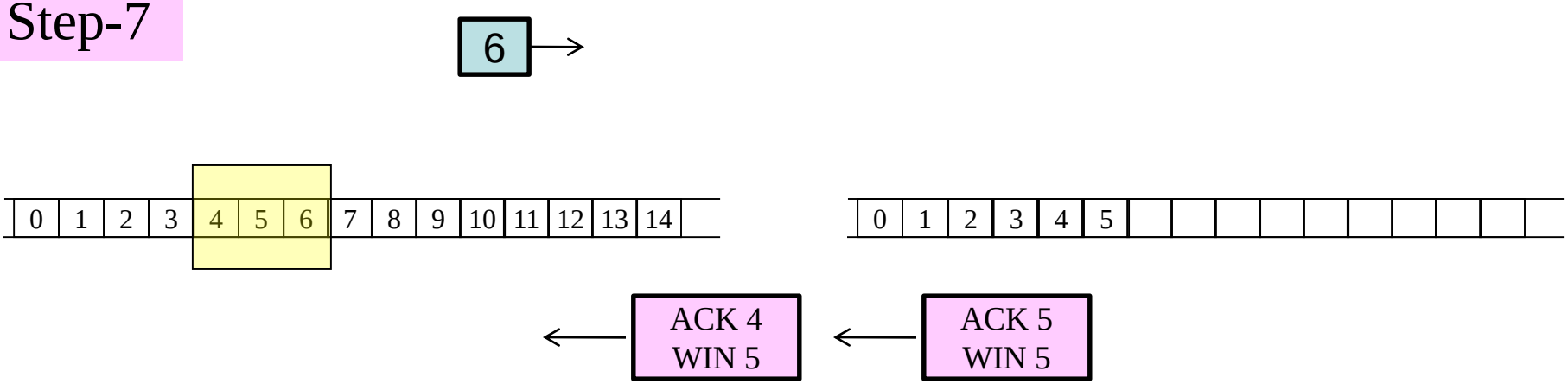




A has ACK of frame 2 and advances the left side of its window by one. However, frame 2's ACK had the window size 3 therefore A cannot advance the right side of its window.

B Receives frame 5 and sends the corresponding ACK with window size of 5.

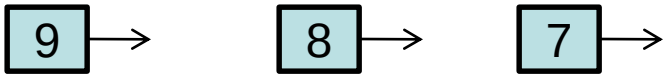
## Step-7

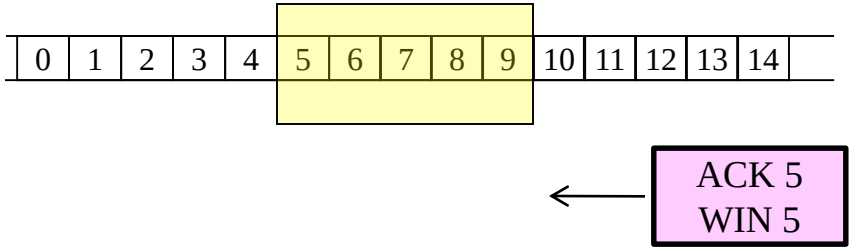


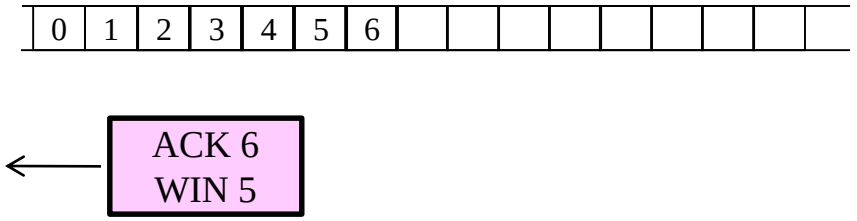

A has ACK of frame 3 and advances both the left and right side of its window by one. This enable the window size of 3 hence A sends frame 6.

B has not received any frame hence sends no ACK .

# Step-8



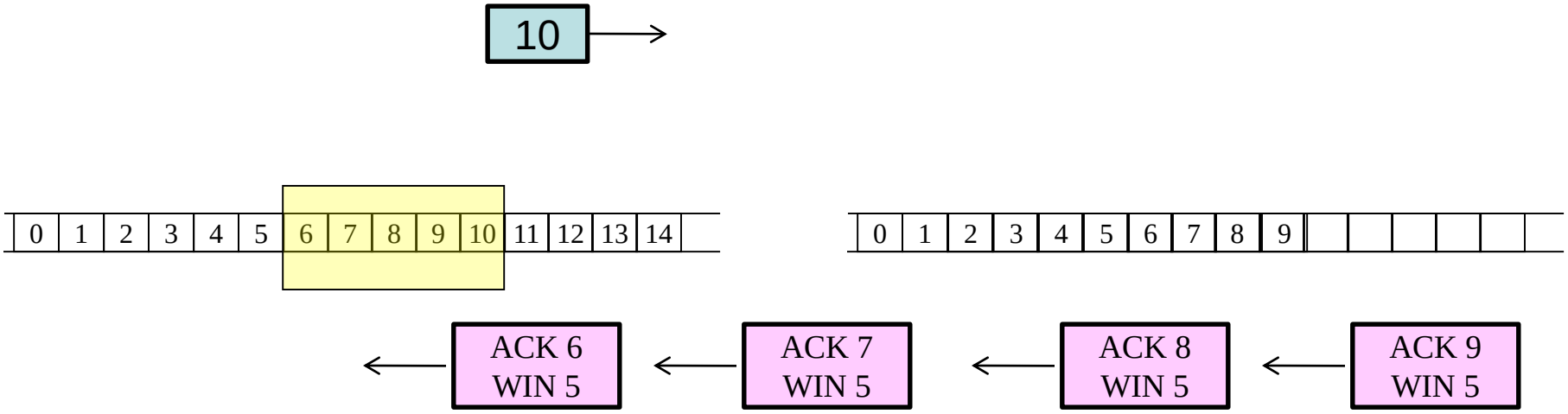




A has ACK of frame 4 and advances the left side by 1 and right side by 3 since ACK-4 provides window size of 5. This permit to send A 3 more frames 7, 8 and 9.

B receives frame 6 and sends ACK with window size of 5.

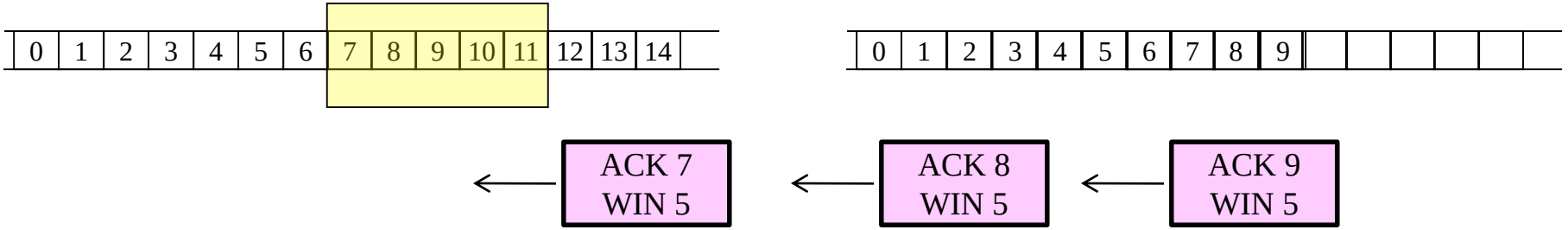
# Step-9




A receives ACK of frame 5 which has WIN 5. Thus A advances the left and right side of its window by 1 and transmits frames 10.

B receives frame 7, 8, 9 and sends their corresponding ACK with window size of 5.

# Step-10


A receives ACK of frame 6 maintains its window size of 5, and transmits frame 11.

B has not received any frame at this stage and hence sends no ACK.

# The Medium Access Control (MAC) Sublayer

The MAC sublayer provides the protocol that define the manner in which nodes share the single physical transmission medium. Actually few physical channels (for example each time slot of a TDMA frame is called physical channel) are shared by huge number of users. The name says it all: Media, Access, Control.

**Three LAN protocol at MAC sublayer:**

- ✓**Random Access Protocol (**some times called *stochastic* or *statistical*)**, for example ALOHA**
- ✓**Controlled Access Protocol (**token passing **)**
- ✓**CSMA/CD and CSMA/CA**

# ✓Random Access Protocol

❑ These protocol employ the philosophy that a node can transmit whenever it has data to transmit. Random access protocols imply contention; in fact, some times we call them contention protocol. Contention is a phenomenon in which more than one entity communicates to do something at the same time.

❑When two or more nodes try to communicate at approximately at the same time, their transmission collide. In LAN terminology we refer to this as a *collision*.

❑For example ALOHA protocol. The efficiency of such protocol is very poor under heavy traffic condition because of frequent collision of packets. Under light traffic condition the throughput of such protocol is high due to low probability of collision. This type of protocol is very simple and can be implemented easily.

## ✓Controlled access protocol:

✓To avoid frequent collision of previous protocol a **token** is first secured by a station prior transmission.

✓Such protocol works efficiently during heavy traffic condition but during low traffic condition a station has to wait unnecessary for its turn. Token ring and token bus network use such protocol.

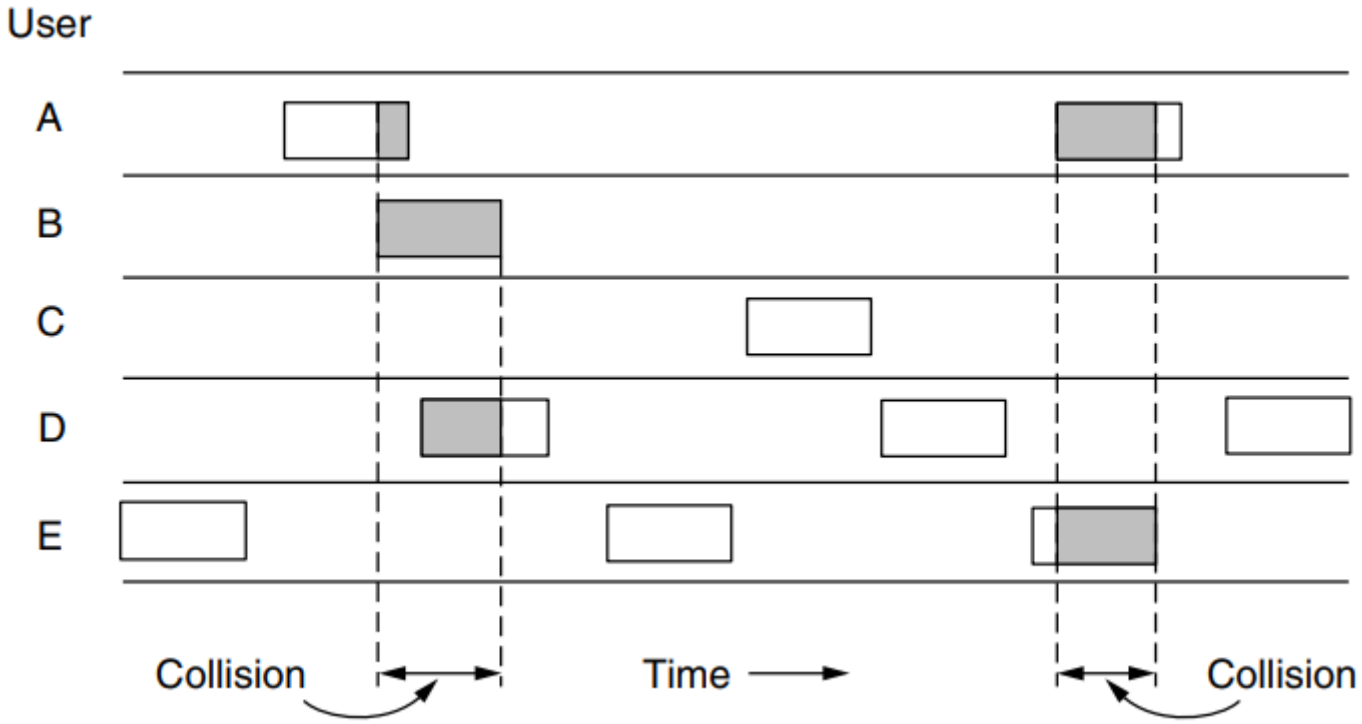
## ✓Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

To minimize the occurrence of collisions, nodes might follow a protocol that requires them first to listen for another node's transmission before they begin transmitting data. We call these types of protocol *carrier sense protocols*. There are two main carries sense protocols:

- ❖CSMA with collision detection (CSMA/CD)
- ❖CSMA with collision avoidance (CSMA/CA)

# Traffic of ALOHA

The simplest multiple access protocol is ALOHA system divided into **pure ALOHA** (there is no time slot and no time synchronization to sent a frame i.e. frames are transmitted in a continuous fashion) and **slotted ALOHA** (discrete version of pure ALOHA where a sender sends frame within a synchronized time slot).



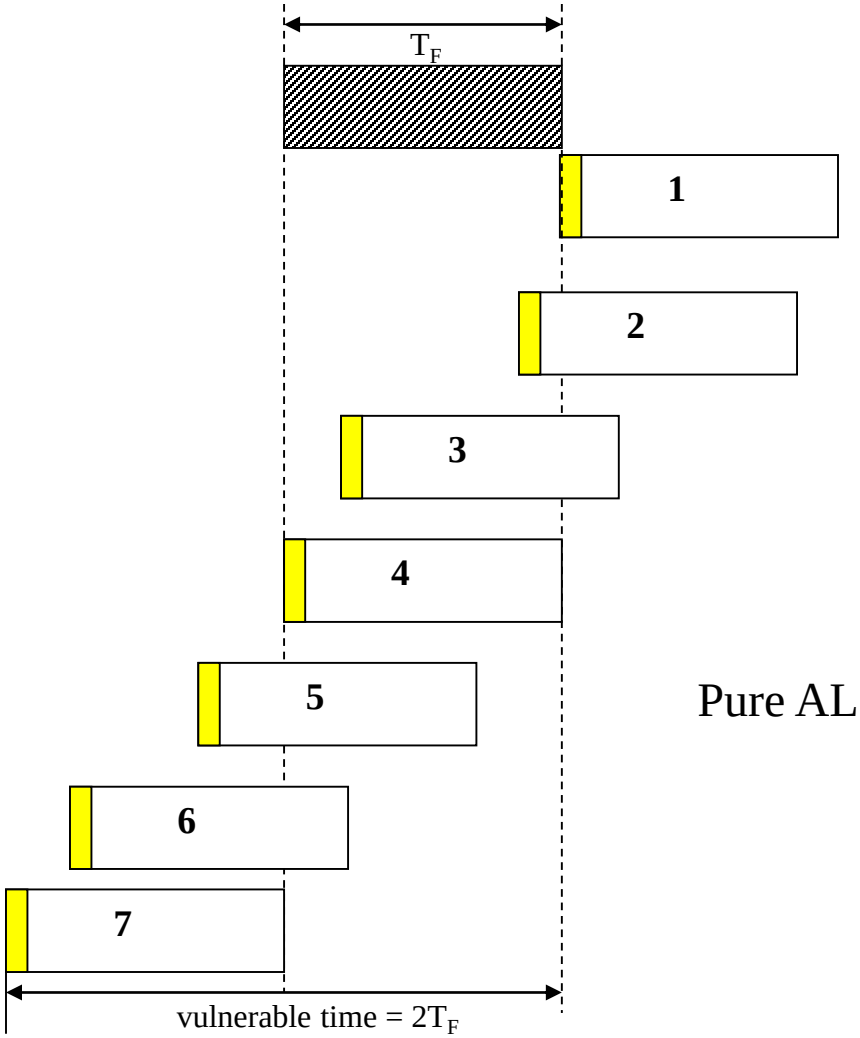

In pure ALOHA, frames are transmitted at completely arbitrary times

✓In **pure ALOHA** users transmit its frame immediately rather than waiting for a beginning of a time slot and wait for a random amount of time to get the acknowledgement.

✓If the sender does not get the **acknowledgement** within that time it sends the frame again. If the transmission time of two ore more nodes overlaps, the reception is garbled therefore all nodes have to retransmit.

✓If only one user transmits or the frame of user is not overlapped by other users then the frame will reach the destination node successfully. The sharing of a common traffic channel by several users in such a way is called **contention system**.

The duration over which two packets overlap called **vulnerable** time. If the duration of a frame is $T_F$ and a user generates the shaded frame like fig.1 then any one of the 7 frames will collide with the shaded frame. From the fig. below we get the vulnerable time of $2T_F$.

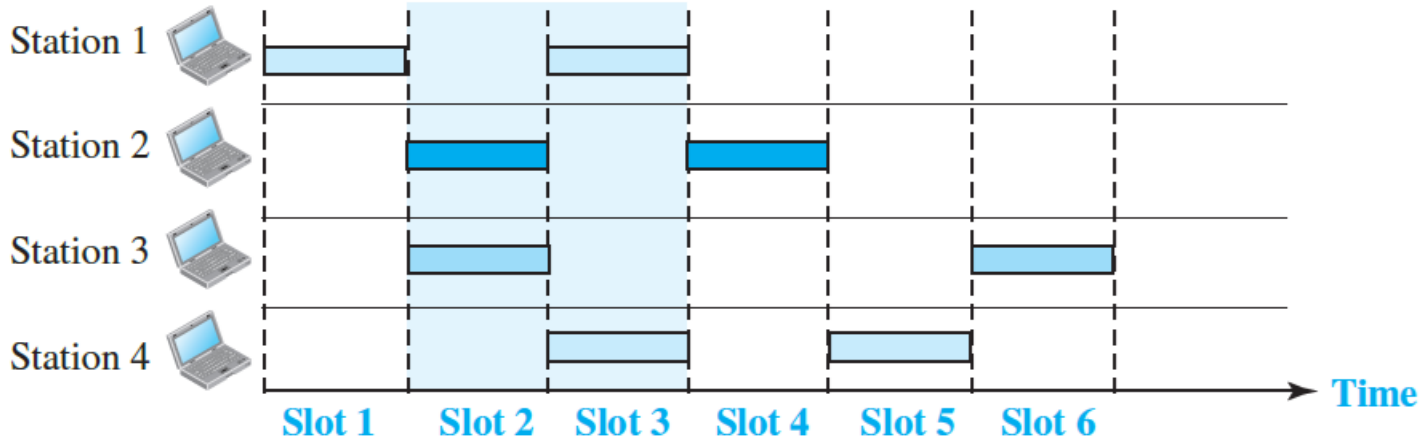



Pure ALOHA

✓In slotted ALOHA all the senders are synchronized with discrete time slot of equal length.

✓Each of the sender sends the frame within a timeslot where the staring instant of the frame is synchronized with the initial position of the time slot. Whenever a user has to end data it has to wait for the beginning of the next time slot.

✓Thus the slotted ALOHA can be considered as the discrete version of pure (continuous) ALOHA.



- ✓ Let the length of a frame is $L$ bits and the transmission rate of a node is $R$ bps (bits/sec). Therefore the length of a frame will be, $L/R = T_F$ seconds. If $S$ frame arrives within a frame duration ($T_F$ seconds) then frame arrival rate, $\lambda_a = S/ T_F$.
- ✓ When the number of channel $n = 1$ then the network can handle frames for the range: $1 > S > 0$ because if $S > 1$ the frames will experience huge collision.

- ✓ If we include the retransmission of frame over the duration $[0, T_F]$, the total number of arrival of frame:

$G$ = no. of newly generated frames + no. of retransmitted of frames because of previous collision

$= S + S_r$.

Now the total frame arrival rate,

$\lambda_t = \frac{S+S_r}{T_F} = S/ T_F + S_r/ T_F = \lambda_a + \lambda_r$ frames/sec.

Now the probability of arrival of k-frames during [0, $T_F$] can be obtained using **Poisson's pdf** like,

$$P_k = \frac{A^k}{k!} e^{-A} = \frac{(\lambda_t T_F)^k}{k!} e^{-\lambda_t T_F}$$

$$P_k = \frac{G^k}{k!} e^{-G}$$

Offered traffic, $A = \lambda_t T_F = G$

Again the probability of no collision is equal to the probability of no arrival within the vulnerable period [0, $2T_F$] can be expressed as,

$$P_0 = \frac{(\lambda_t 2T_F)^0}{0!} e^{-\lambda_t 2T_F} = e^{-2G}$$

Therefore the throughput of the system,

$\overline{X}$ = Offered traffic × the probability of no collision

$= (\lambda_t . T_F) . P_0 = G e^{-2G}$

Fraction of offered traffic without collision is the carried traffic.

Let us now determine the maximum throughput and the corresponding value of G.

$$\frac{d\bar{X}}{dG} = -2Ge^{-2G} + e^{-2G} = 0$$

$$\Rightarrow e^{-2G}(-2G + 1) = 0$$

$$\Rightarrow (-2G + 1) = 0$$

$$\Rightarrow G = 1/2$$

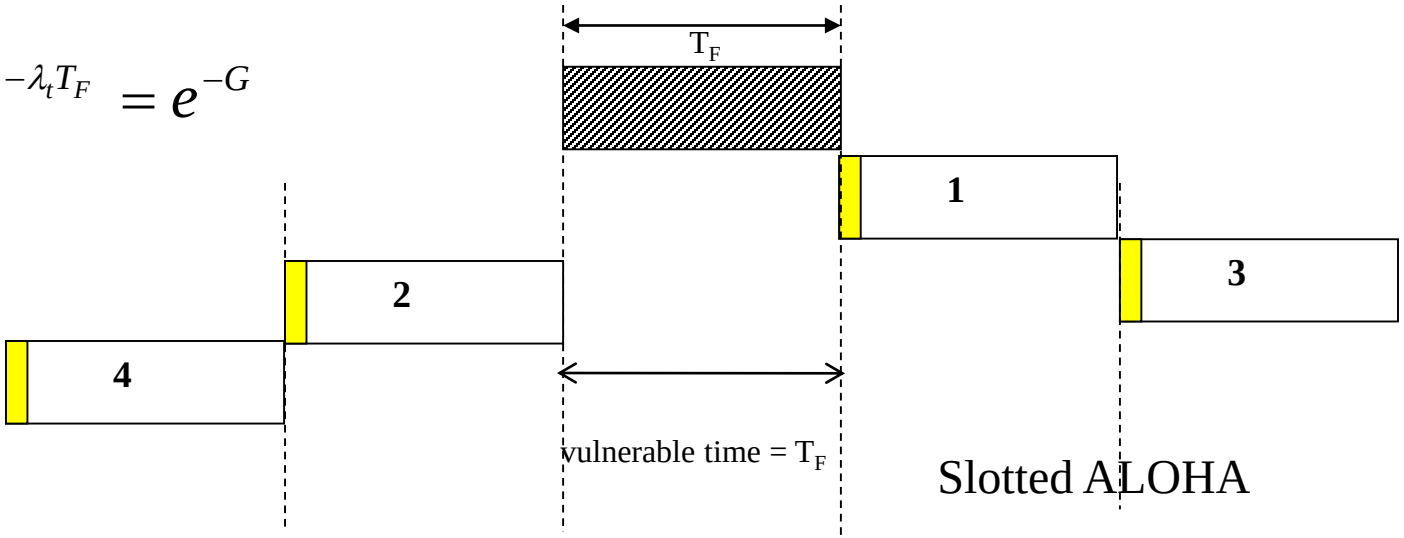
Offered traffic, $A = \lambda_t . T_F = G$

The maximum throughput, $\bar{X} = \frac{1}{2} e^{-2\frac{1}{2}} = \frac{1}{2e} = 0.184_{\max}$

The maximum channel utilization of pure ALOHA is only 18.4%

For slotted ALOHA the vulnerable period is $T_F$, therefore the probability of no collision,

$$P_0 = \frac{(\lambda_t T_F)^0}{0!} e^{-\lambda_t T_F} = e^{-G}$$




Slotted ALOHA

Therefore the throughput of the system,

$$\bar{X} = Ge^{-G}$$

Let us now determine the maximum throughput and the corresponding value of G.

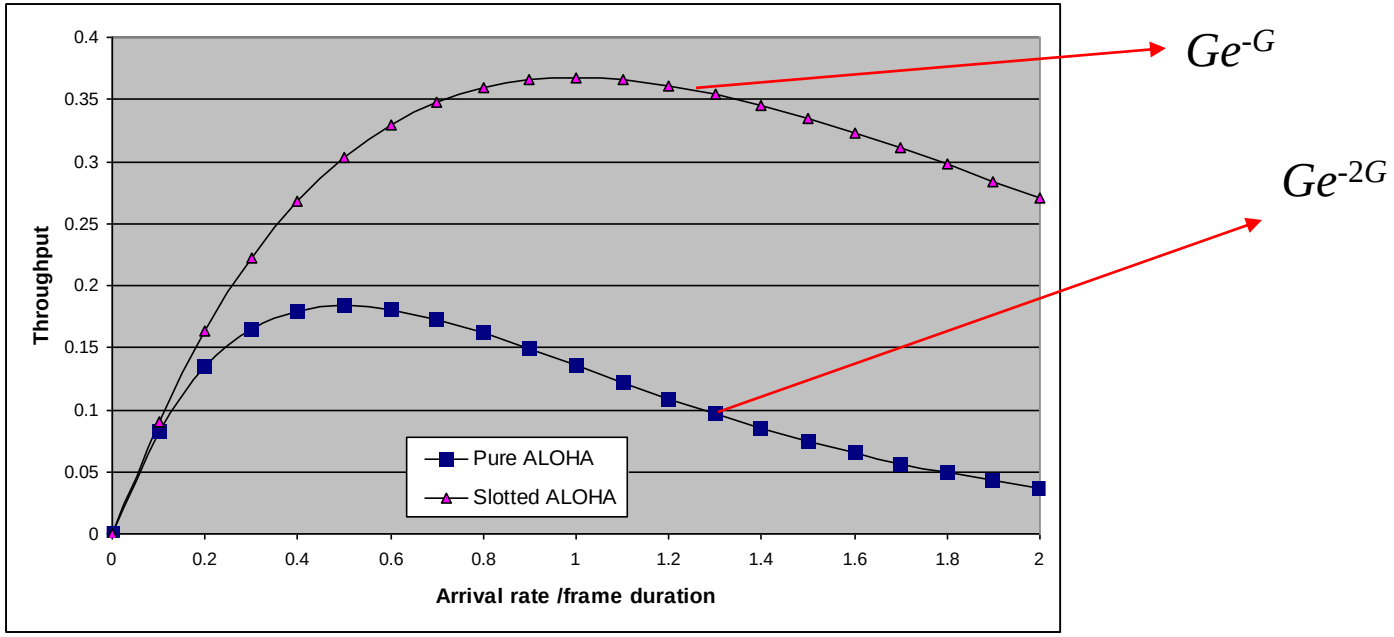
$$\frac{d\bar{X}}{dG} = -Ge^{-G} + e^{-G} = 0 \qquad \Rightarrow e^{-G}(-G+1) = 0$$

$$\Rightarrow (-G+1) = 0 \qquad \Rightarrow G = 1$$

The maximum throughput,

$$\overline{X}_{\max} = e^{-1} = \frac{1}{e} = 0.3684$$

The maximum channel utilization of slotted ALOHA is 36.8%. The profile of pure and slotted ALOHA is shown in fig. below.



Example-1

A pure ALOHA system uses 60 kbps channel where on an average each terminal generates frame of 1000 bits, where each user on average transmits one frame on every 20 second. How many terminal the system can accommodate? Repeat the job for slotted ALOHA case.

Ans.

The duration of a packet, $T_F$ = 1000bits/($60\times10^3$)bits/sec

= $16.67\times10^{-3}$ sec = 16.67ms

The offered traffic by each terminal,

$G = \lambda.T_F = (1/20)\times 16.67\times10^{-3} = 8.335\times10^{-4}$ Erlangs/users

Let the number of users are $N$, therefore the total offered traffic,

$G\times N$ = 1/2e = 0.184 = maximum carried traffic (the maximum % of utilization of the channel)

$8.335\times10^{-4}N = 0.184$

N = 220.75 = 221

For slotted ALOHA case,

$8.335\times10^{-4}$N = 0.3684

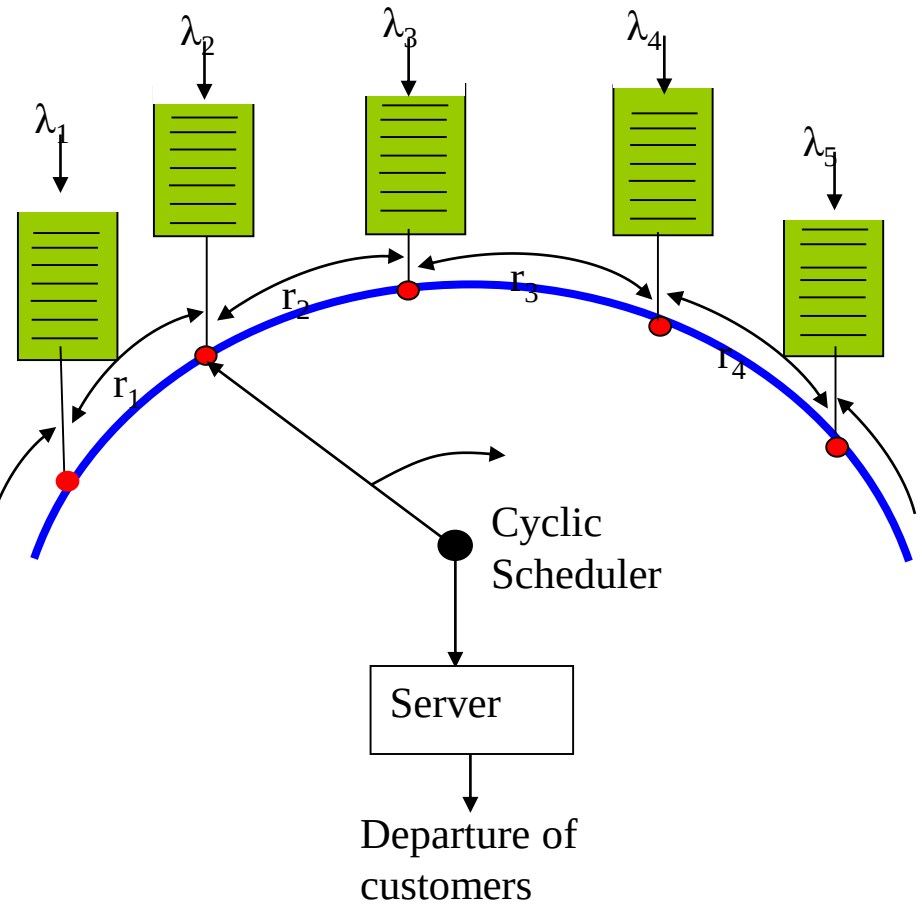
N = 441.99 = 442

# Example: Token Ring

✓In token ring LAN a special packet called token (permission of access the ring) circulates among a number of terminals in a ring configuration shown in fig. of next slide.

✓Terminals get the opportunity to seize the token sequentially, once the terminal gets the token, the terminal converts it to a busy packet with destination address.

✓After receiving the packet the destination node sends the acknowledgment to the source node. Receiving the acknowledgment the sending node generates a new token to be used by another station.




Let the message arrival rate at queue according to independent Poisson's process with mean rates,

$$\lambda_1 = \lambda_2 = \lambda_3 = \cdots \quad \cdots \quad \cdots = \frac{\lambda}{N}$$

Propagation time between adjacent node,

$$r_1 = r_2 = r_3 = \cdots \quad \cdots \quad \cdots = \frac{\tau}{N} = r$$

;where $r_i$ is the propagation time (or walking time) from queue $i$ to $i$+1 and the total propagation delay of the ring,

$$\tau = \sum_{i=1}^{N} r_i$$

The mean waiting time (queuing delay) at sending station,

$$E[W] = \frac{\delta^2}{2r} + \frac{Nr(1-\rho/N)}{2(1-\rho)} + \frac{\rho E[S^2]}{2(1-\rho)E[S]}$$

where $\delta^2$ is the variance of propagation time

The traffic intensity ,

$$\rho_i = \lambda_i E[S_i]$$

$E[S_i]$ is the mean service time of message from queue *i* follows the general distribution .

The mean service time (packet length in seconds) expressed as,

$E[S] = (E[L_p] + L_h)/C = 1/\mu = \rho/\lambda$

where $L_p$ and $L_h$ are the mean packet (message part) length and header length in bits and $C$ is the channel capacity in bps.

The average propagation delay, $E[T_p] = \tau / 2$

Where $\tau$ is the round trip propagation delay.

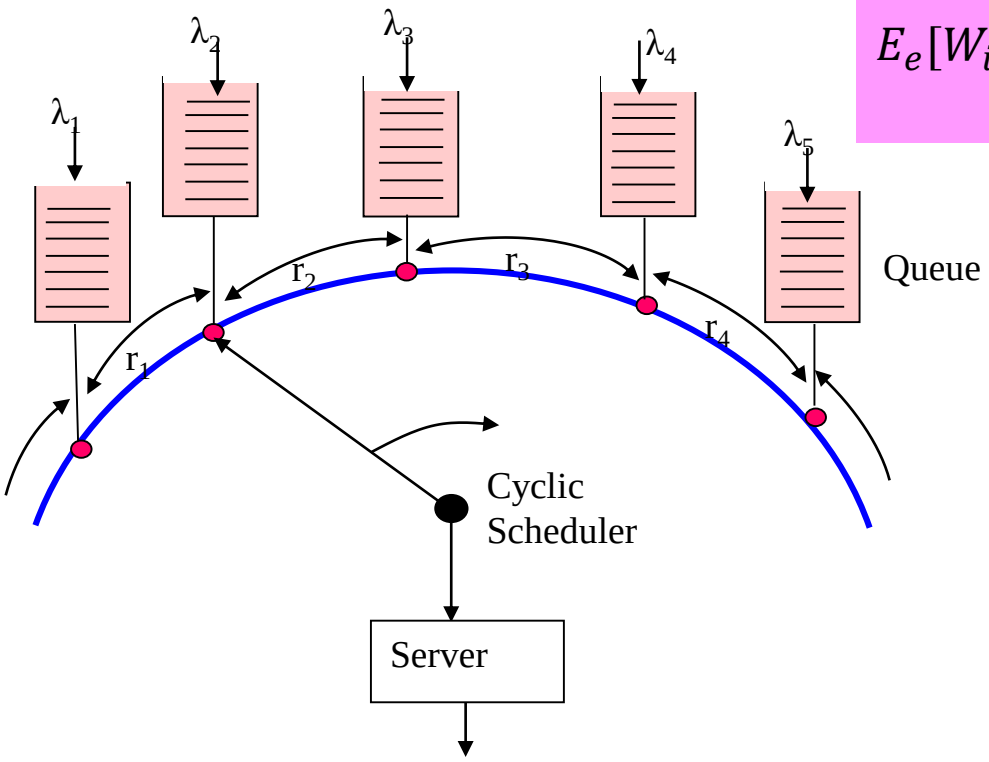
The propagation time *Tp* is the time elapsed from the beginning of the transmission of the message until the arrival of first bit of the message at the destination. The mean transfer delay,

$E[D] = E[W]+E[S]+E[Tp]$.

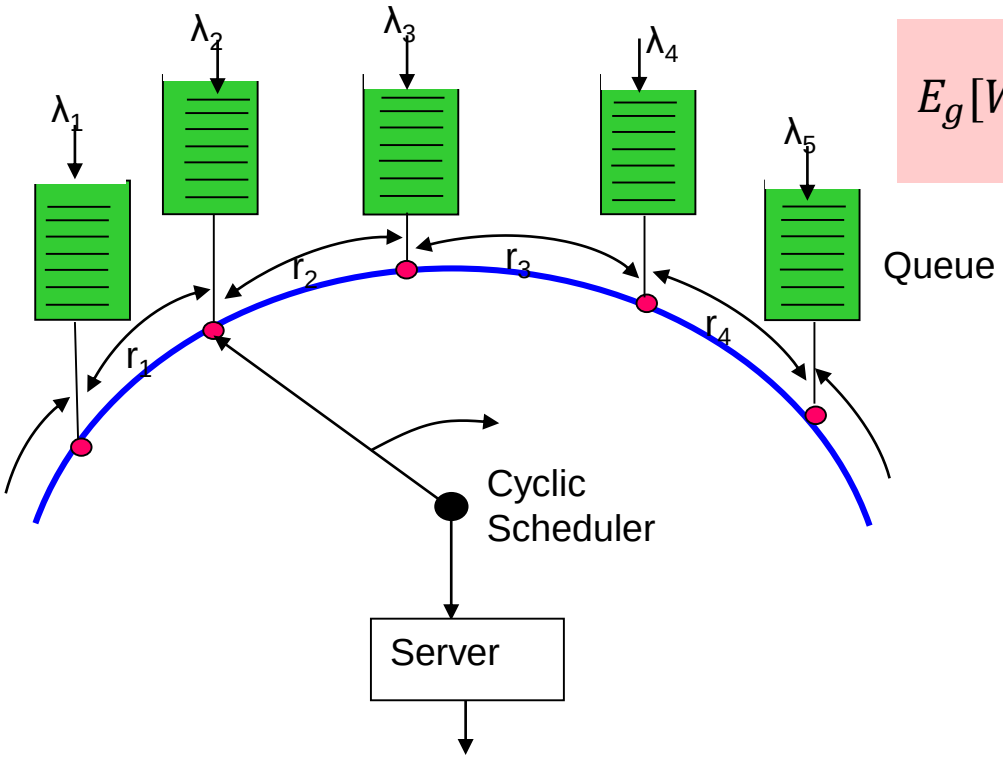
When a station captures a token its serves the queue in one of the following three types of service.

(a) The server serves all the packets (including those received during the transmission) in the queue (no packets of a customer is left) called exhaustive service. The mean waiting time of **exhaustive service** is expressed as,

$$E_e[W_i] = \frac{\delta^2}{2r} + \frac{Nr(1-\rho/N)}{2(1-\rho)} + \frac{\rho E[S^2]}{2(1-\rho)E[S]}$$



(b) The server serves only its queued packet which arrived just before capturing the token. In other words, the token is made free after transmitting the packets which were waiting before the transmission began i.e. the packets in the queue arrived after getting the token will not be served. This type service is called **gated service**. The mean waiting time of gated service is expressed as,




$$E_g[W_i] = \frac{\delta^2}{2r} + \frac{Nr(1+\rho/N)}{2(1-\rho)} + \frac{\rho E[S^2]}{2(1-\rho)E[S]}$$

(c) The server serves a limited number of packet $k$ may be less than the number of waiting packets in the queue called **limited service**. The mean waiting time of limited service is expressed as,

$$E_l[W_i] = \frac{\delta^2}{2r} + \frac{Nr(1+\rho/N) + N\lambda\delta^2}{2(1-\rho-N\lambda r)} + \frac{\rho E[S^2]}{2(1-\rho-N\lambda r)E[S]}$$

Example-1

A token-ring has a total propagation delay of 20μs, a channel capacity of 1Mbps and 50 stations each of which generates a Poisson traffic and has a latency of 1 bit. For traffic intensity of 0.6, calculate (a) The switchover time/propagation time (b) The mean service time (c) The message arrival rate per station (d) The average delay for exhaustive\, gated and limited service discipline. Assume 10 bits for overhead and 500 bits average packet length, exponentially distributed.

(a) The switchover time, $r = \tau/N = 20/50 = 0.4\mu s$

(b) The mean service time, $E[S] = (E[L_p] + L_h)/C = (500+10) / 10^6 = 510\ \mu s$

(c) Now, $\rho = \lambda E[S] \Rightarrow \lambda = \rho / E[S]$ . Therefore arrival rate per station is, $\lambda_i = \rho / NE[S] = 0.6/(50\times510\times10^{-6}) = 23.53$ frame/sec

(d) For exponentially distributed packet length, $E[S^2] = 2E^2[S] = 2\times(510\times10^{-6})^2\ s^2 = 52.02\times10^{-8}\ s^2$

Since, $r_1 = r_2 = r_3 = \cdots \quad \cdots \quad \cdots = \frac{R}{N} = r$ then the variance $\delta^2$ of switchover time is zero.

**For exhaustive service case,**

$$E_e[D] = \frac{\delta^2}{2r} + \frac{Nr(1-\rho/N)}{2(1-\rho)} + \frac{\rho E[S^2]}{2(1-\rho)E[S]} + E[S] + \tau/2$$

$$=0+\frac{50\times0.4\times10^{-6}(1-0.6/50)}{2(1-0.6)} + \frac{0.6\times52.02\times10^{-8}}{2(1-0.6)\times510\times10^{-6}} + 510\times10^{-6} + 20\times10^{-6}/2$$

$=1.3097\times10^{-3}$s $=1.3097$ms

For gated service case,

$$E_g[D] = \frac{\delta^2}{2r} + \frac{Nr(1+\rho/N)}{2(1-\rho)} + \frac{\rho E[S^2]}{2(1-\rho)E[S]} + E[S] + \tau/2$$

$$= 0 + \frac{50\times0.4\times10^{-6}(1+0.6/50)}{2(1-0.6)} + \frac{0.6\times52.02\times10^{-8}}{2(1-0.6)\times510\times10^{-6}} + 510\times10^{-6} + 20\times10^{-6}/2$$

=1.3103ms

For limited service case,

$$E_l[D] = \frac{\delta^2}{2r} + \frac{Nr(1+\rho/N) + N\lambda\delta^2}{2(1-\rho-N\lambda r)} + \frac{\rho E[S^2]}{2(1-\rho-N\lambda r)E[S]} + E[S] + \tau/2$$

$$=0 + \frac{50\times0.4\times10^{-6}(1+0.6/50)+0}{2(1-0.6-50\times(23.53\times50)\times0.4\times10^{-6})} + \frac{0.6\times52.02\times10^{-8}}{2(1-0.6-50\times(23.53\times50)\times0.4\times10^{-6})510\times10^{-6}} + 510\times10^{-6} + 20\times10^{-6}/2$$

$=1.3597\times10^{-3}$s =1.3597ms

Here we found, $E_e[D] < E_g[D] < E_l[D]$.

# Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- ✓ In carrier sense multiple access with collision detection (CSMA/CD), a node starts its transmission after sensing that there is no carrier in the bus therefore collision of packets from simultaneous users are reduced.

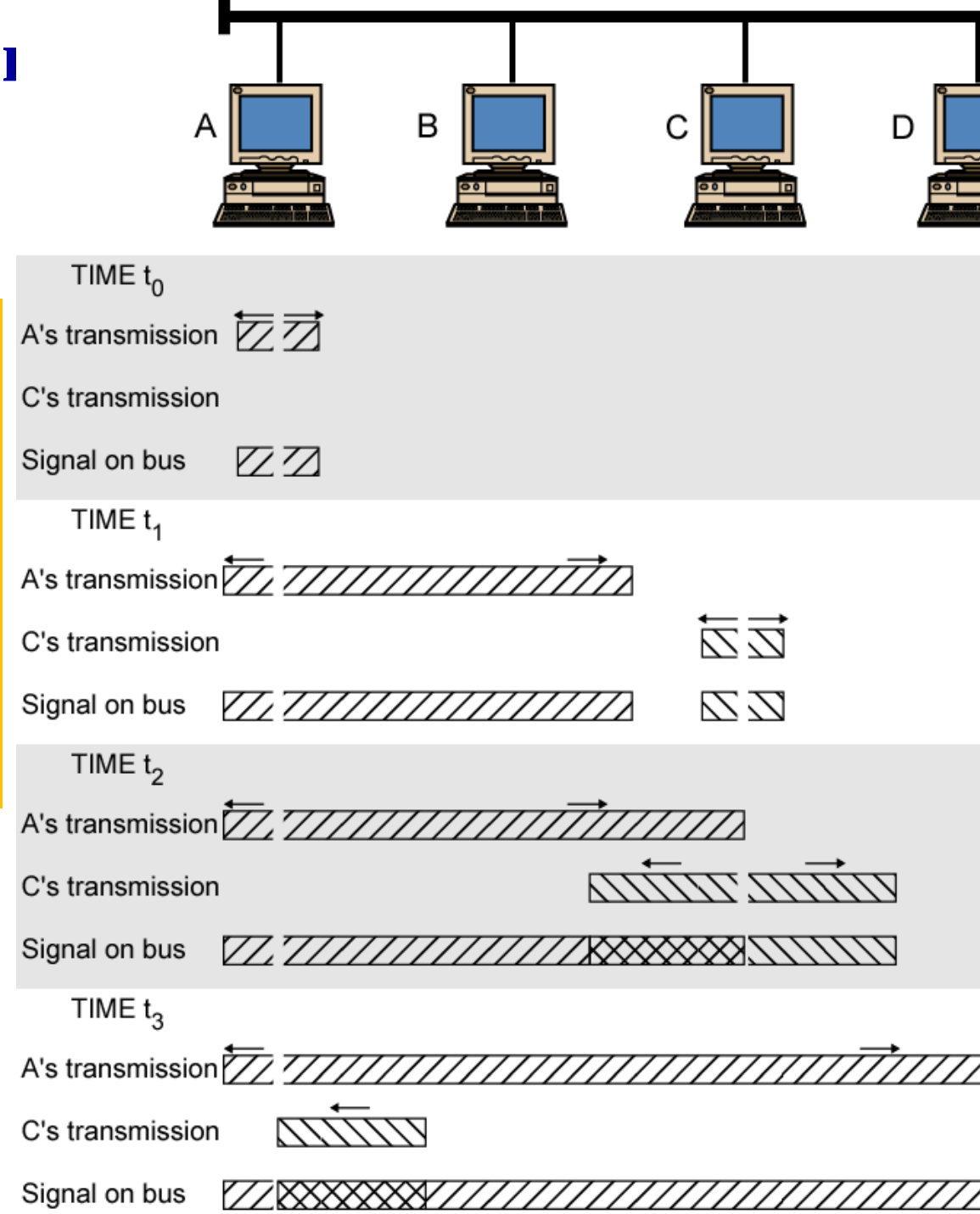
- ✓ If the node finds the bus busy then it waits for a random amount of time and re-sense the channel. If a node experiences collision during transmission it stops the transmission and wait for a random amount of time for retransmission.

# CSMA/CD Operation

In wireless communication the situation is cumbersome since the received power is too small compared to transmitted power. Therefore collision avoidance (CA) is more realistic for wireless network instead of collision detection (CD). Hence wireless network use CSMA/CA.

C detects a collision!

A detects a collision!



Three different types of protocols are used under CSMA/CD

(a) 1-persistent CSMA

(b) non-persistent CSMA

(c) p-persistent CSMA.

✓The first one is called **1-persistent CSMA**. Under this scheme if a node/station finds the channel is busy, it waits until the channel becomes idle. When the station detects the channel is idle it transmits with probability of 1. If a collision occurs the station waits for a random amount of time and repeats the job.

✓Under **non-persistent CSMA** case if the station finds the channel is busy it does not sense the channel continuously to seize it just after becoming free. Instead it waits for a random amount of time and starts the job. The scheme is less greedy than the previous one. It increases the utilization of channel but delay will be longer. The throughput of such scheme is better than the previous one under heavy traffic condition.

- ✓ The thirds protocol is ***p*-persistent CSMA** applied for slotted channel. Under this scheme a station sensing the idle channel transmits with probability $p$ and defers with probability $q = 1-p$ until next slot. If the next slot is also idle it performs the similar job with probabilities $p$ and $q$. Therefore a station either transmits or another station starts transmission.

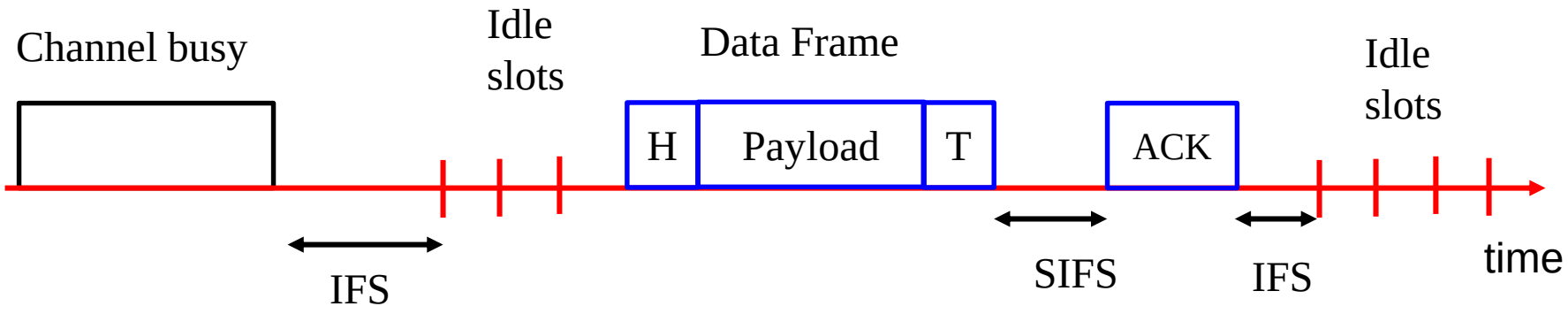
- ✓ The behavior of the deprived station is like that it may face the collision so it will wait for a random amount of time.

# Carrier Sense Multiple Access/ Collision Avoidance (CSMA/CA)

- The wireless LAN system cannot detect collisions because the power of the transmitting device is much stronger than the its received signal power.
- In this situation collision detection is not practical, it makes sense to try to devise a system that can help prevent collisions. Thus the CA is CSMA/CA refers to ‘collision avoidance’.
- In wired communication both the transmitted and received signals are equally strong therefore collision can be detected easily, whereas situation is not favorable in wireless link, there collision avoidance (CA) is used in wireless network instead of collision detection (CD).

➢If the sender does not receive an ACK then the device assumes that a collision has occurred. Sender usually uses **binary exponential backoff**.

# Binary Exponential Backoff Algorithm



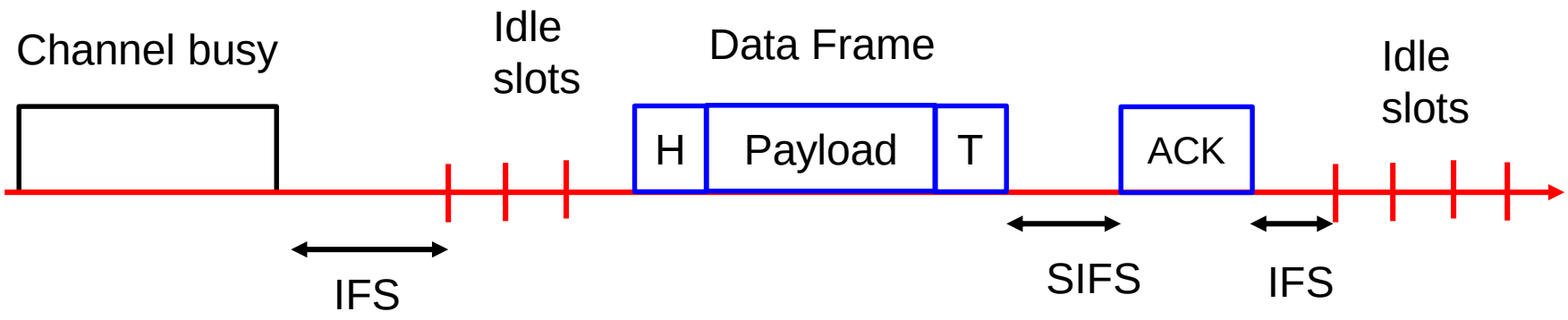

To reduce the packet dropping probability or to enhance throughput of wireless LAN exponential binary backoff algorithm is widely used. The access method of MAC protocol of IEEE 802.11 based on **exponential binary backoff algorithm** can be explained with the following steps.

**Step: 1**

The transmitting node first senses the status of the channel. If the channel is found busy then the Tx node continues to monitor the channel.

**Step:2**

If the channel is found idle for a fixed duration know as IFS (Inter-frame Space), the Tx chooses a random number according to the binary exponential back off algorithm. The random number is used as a back off timer.

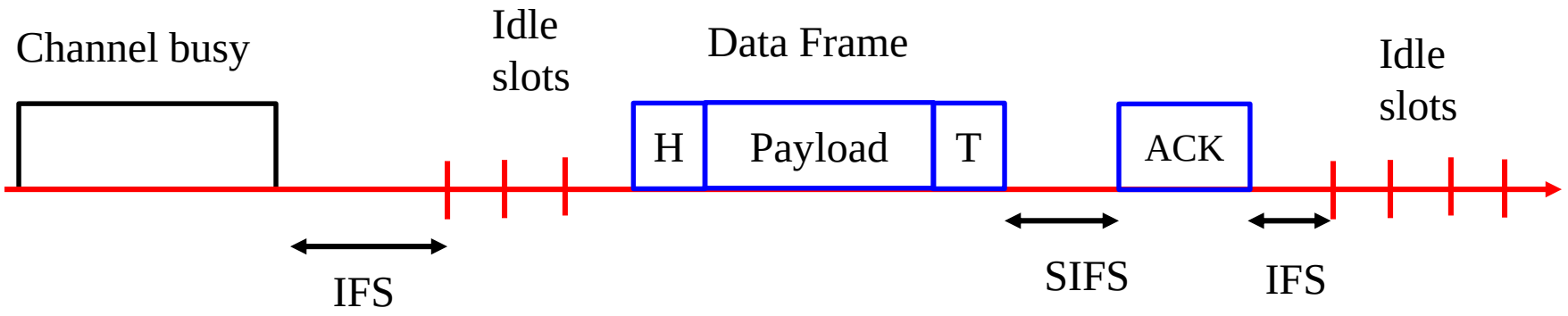



**Step:3**

Time immediately after the IFS (Inter-frame Space) is slotted known as idle slots where the duration of a slot is considered as the sum of the time required to sense a station and to switch the Tx from sensing / listening mode to transmitting mode.

**Step:4**

Elapsing of each idle slot the back off timer is decreased by one. If the channel is found busy before the back off timer reaches to zero then repeat the steps 1 to 3. The transmission of data from begins only if the back off timer reaches to zero.




**Step:5**

✓To determine whether a data frame transmission is successful, after its completion, a positive acknowledgement (ACK) is transmitted by the receiver. ACK is transmitted after a short interframe space (SIFS) period upon receiving the entire data frame successfully.

✓If ACK is not detected within an SIFS period after the completion of the data frame transmission, the transmission is assumed to be unsuccessful, and a retransmission is required.